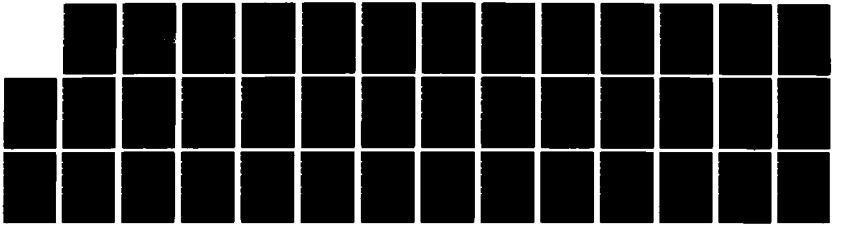
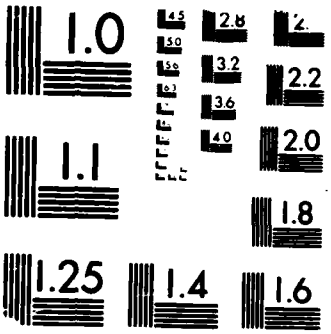
AD-A191 541 CALCULATING THE PRINCIPAL VIEWS OF A POLYHEDRON(U) 1/1
ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE N WATTS
DEC 87 TR-234 N00014-82-K-0193

UNCLASSIFIED F/G 12/1 NL





Calculating the Principal Views of a Polyhedron

Nancy Watts
Department of Computer Science
The University of Rochester
Rochester, NY 14627

TR 234
December 1987

Department of Computer Science
University of Rochester
Rochester, New York 14627

(3)

Calculating the Principal Views of a Polyhedron

Nancy Watts
Department of Computer Science
The University of Rochester
Rochester, NY 14627

TR 234
December 1987

Abstract

This report examines the "principal views" of polyhedra whose images are line drawings representing the images of edges. Several definitions for useful equivalence relations on the set of viewpoints are compared. An algorithm for enumerating the faces in each of the principal views of a convex polyhedron under perspective projection is given; this algorithm admits non-trihedral vertices. An implementation of the algorithm is described.

This research was supported in part by the Office of Naval Research Defense Advanced Research Projects Agency under Contracts N00014-82-K-0193 and N00014-84-K-0655. The Xerox Corporation University Grants Program provided the equipment used in preparing the paper.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Tr 234	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Calculating the Principal Views of a Polyhedron		5. TYPE OF REPORT & PERIOD COVERED technical report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Nancy Watts		8. CONTRACT OR GRANT NUMBER(s) N00014-82-K-0193 N00014-84-K-0655
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dept. of Computer Science 734 Computer Studies Bldg. University of Rochester, Rochester, NY 14627		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA / 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE December 1987 13. NUMBER OF PAGES 36
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. *(of this report)* unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution of this document is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

principal view
characteristic view
aspect graph
object recognition
polyhedral scene analysis
face-labeled line drawings

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report examines the "principal views" of polyhedra whose images are line drawings representing the images of edges. Several definitions for useful equivalence relations on the set of viewpoints are compared. An algorithm for enumerating the faces in each of the principal views of a convex polyhedron under perspective projection is given; this algorithm admits non-trihedral vertices. An implementation of the algorithm is described.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

1. Introduction

Humans recognize a multitude of objects with great ease despite the fact that for every object there are uncountably many distinct possible images. How is this done? Robots need to be programmed to do the same task, perhaps in simplified form. How could that be done? It has been suggested that a sort of catalogue of finite sets of "topologically distinct" views of each object could be made for use in this recognition problem. What does topologically distinct mean? How could such views be calculated for a particular object? How many are there?

The focus of this report is to explore these questions in a simplified situation: the objects are polyhedra and the images are line drawings representing the images of edges. An algorithm for enumerating the faces in each of the "principal views" of a convex polyhedron under perspective and orthographic projection is given; this algorithm admits non-trihedral vertices. An implementation of the algorithm is described. The algorithm has time complexity $O(n^4)$ where n is the number of faces. This is less efficient than an $O(n^3)$ algorithm reported in Edelsbrunner, O'Rourke, and Seidel (1983). However, the algorithm given here provides an explicit description of the principal views and any algorithm which does so must have time complexity $O(n^4)$. For applications, an explicit description is likely to be required; the algorithm given here has the advantage of relative simplicity.

2. Basic Definitions

Let P be a polyhedron. An image of P might be formed either by perspective projection or orthographic projection. Under perspective projection, there are distinct images of P for a viewpoint at all points in three-dimensional space outside the convex hull of P. Call these points the viewing space for perspective projection. Under orthographic projection, only the direction of view matters. Each direction can be associated with a point on the unit sphere; this sphere will be called the viewing space for orthography. Alternately, since orthographic projection amounts to perspective projection with viewpoints at infinity, the orthographic viewing space can be thought of as a sphere with infinite radius. In either case, an equivalence relation must be defined on the viewing space to differentiate regions with "topologically distinct" images.

In the literature, several terms are used in the process of describing equivalence relations on views of polyhedral objects. The terms *principal view*, *characteristic view*, *visual potential*, and *aspect graph* refer to more or less to the same notions. In this paper, the term principal view will be used. At the end of the report a review of relevant literature is given.

In this report, the term *polyhedron* refers to a solid which is bounded by planar faces. The boundary of each face consists of line segments called edges and the boundary of each face must form a single closed polygon. In the general case, these

polygons and indeed the polyhedron need not be convex. The intersections of three or more faces that lie on the boundary of the solid are called its vertices.

3. Defining the Equivalence Relation

Let us consider some definitions given in the literature.

In the aspect graph definition given by Plantinga and Dyer (1986) the equivalence relation is defined as follows: At any particular point in the viewing space, a subset of the faces of P is visible. Two points q_1 and q_2 are equivalent if and only if the same set of faces are visible and there is a continuous path from q_1 to q_2 in the viewing space in which exactly these faces are visible. The resulting equivalence classes are sets of points (which should be thought of as regions in space) in which exactly the same subset of the faces are visible. Because of the continuous path requirement, two points from which the same faces are visible may or may not be in the same equivalence class. These sets of visible faces become labels of nodes in the aspect graph. There is an edge between two nodes in this graph just in case the associated regions are adjacent to each other, i.e., there is a path from some point in one of the regions to a point in the other region which passes through no other regions. Note that under this definition, nothing is stated about the visibility of edges or vertices in the image. Two points could be equivalent but be such that the edge between two visible faces is visible at one of the points but not the other. Also, a face might be completely visible at one point but partly obscured at another equivalent point. Examples of complex non-convex polyhedra can be constructed where the views from different points in the same equivalence class seem intuitively to be quite different.

Consider an example of such a situation as sketched in Figure 1. Three views of a solid polyhedron, which consists of two blocks glued together rather haphazardly, are shown. Since the same faces are visible in all of these views, and it is possible to construct a path between any pair of the viewpoints corresponding to them such that the same faces are visible, they would be equivalent under the Plantinga and Dyer definition. But there are several differences. For example, the image of the top face of the larger block appears as two disconnected pieces in two of the views and one in the third. Does this conform to our idea of topological equivalence?

The notion of aspect graph was first introduced by Koenderink and van Dorn (1979). It is also used by Werman, Baugher, and Gualteri (1986) and by Stewman and Bowyer (1987). The last two papers do not mention the continuous path requirement. However, they are only concerned with the convex case where such a requirement is redundant.

In the characteristic view definition of Chakravarty and Freeman (1982) and Chakravarty (1980) there is a similar equivalence relation. Here, however, for two points to be equivalent, it is the visible edges and vertices together with their connectivity relationships that must be the same rather than the set of visible faces. Again, there must also be a continuous path between the two points such that the

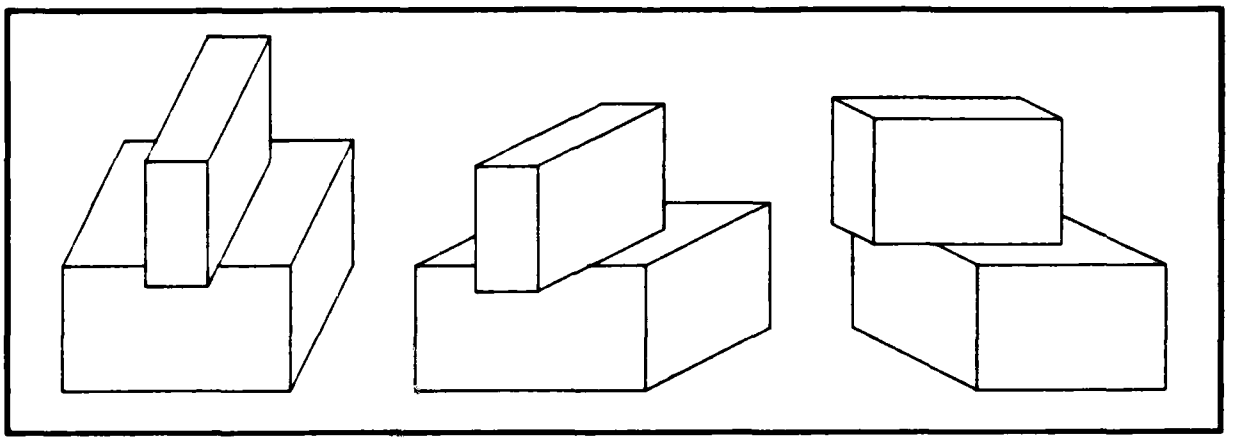


Figure 1

same edges, vertices, and relationships are visible along the path. Unfortunately a precise definition of the term "connectivity relationship" is not made. The equivalence relation does seem to be closer to an intuitive idea of sameness. (It should be noted that the definition is intended to apply to other solid objects, not just polyhedra.)

One perplexing question which arises is this: suppose we have a complex polyhedron as shown in Figure 2. One can imagine traveling in a path so that the small pyramid appears and disappears behind the columns. There will be three unconnected regions in space where it has been hidden by a different column. The line drawings associated with each of these regions will be essentially the same, but there is no path between points in two different regions which stays within these regions. Are the points in two different regions equivalent or not? A continuous path requirement causes them to fail to be equivalent even though the images seem to be topologically the same.

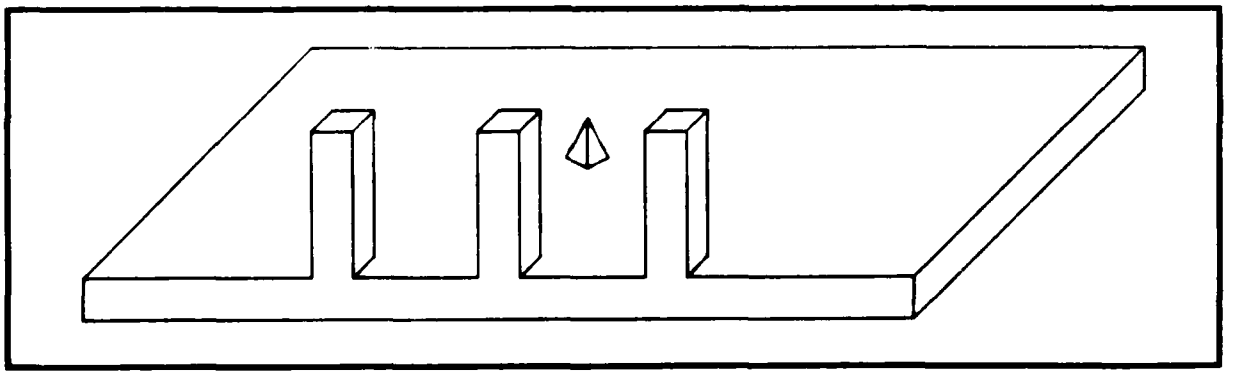


Figure 2

What kind of equivalence relation is appropriate? Before addressing this issue, a definition of another kind of equivalence, "topological equivalence of line drawings," is needed. Most of the previous work seems to use an intuitive idea of this notion of equivalence, but it needs to be made precise. In defining equivalence of line-drawing images, we would like to require that equivalent images contain the same faces, edges, and vertices, with the same relationships to each other. Unfortunately, this is not simple to describe carefully.

Suppose a polyhedron with face labels is given and we wish to define what it means for two images to be topologically equivalent. Assign an arbitrary direction to each edge of the polyhedron. For each edge of the polyhedron and each image, construct an edge image description with the following data. Indicate how many pieces (if any) of the edge are visible and whether the end points are visible. Traveling in the assigned direction, for each visible piece give an ordered listing of the adjacent parts (faces and edges) of the polyhedron which are visible to the right and left of the edge, together with the parts visible at the two ends. These descriptions allow us to get the required definition. First, for two images of the polyhedron and an edge e, define the images of e to be equivalent if their descriptions are the same. Now define two images to be equivalent if and only if all the edge images are equivalent.

Another possibility exists for equivalence of line drawings. Suppose there are no labels on the faces in the image and no distinction is wanted between faces that "look the same" in an image. We still have some intuitive notion of topological sameness. For example, in the image of a cube, one, two, or three faces are visible. In some sense, for a cube without face labels, these would be the three possible distinct images. A definition can be made that is similar to the one in the paragraph above; the details are omitted.

4. A Partial Ordering of Equivalence Relations

Assume that a polyhedron is given, possibly with labels on each face, and that a "reasonable" equivalence relation on the points in the viewing space is wanted. Some subset of the following requirements might be used to define equivalent points under the relation:

1. The same sets of faces are at least partially visible at the two points.

2. Face-labeled, line-drawing images obtained with the two points as viewpoints are topologically equivalent.

3. There is a continuous path between the points such that all points on the path are equivalent.

4. Unlabeled line drawings of the images are topologically equivalent.

It is clear that different equivalence relations might be defined. Several possibities seem reasonable. Using the requirements as numbered above the following selections could be used:

R1: 1 and 3 (Plantinga and Dyer)

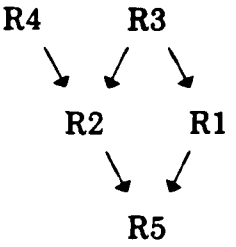
R2: 1 and 2 (The author's intuitive idea)

R3: 1 only

R4: 4 only

R5: 1, 2, and 3 (Chakravarty and Freeman)

These possible definitions are related by a directed graph:



On this graph, an edge from R_i to R_j means that $x\ R_j\ y$ implies $x\ R_i\ y$. Or, in other words, R_j is a kind of refinement of R_i. (That is, the equivalence sets under R_j are subsets of the equivalence sets under R_i.)

Different applications may well make different choices of the fundamental equivalence relation appropriate, so different definitions are probably needed. It would be useful to develop some standard terminology. The R4 definition seems unlike most notions that have been used since it does not involve labels on the faces. In fact, an implicit underlying assumption has usually been made that, somehow, such labels exist and are known. Such a labeling will be assumed here.

The rest of this report concentrates on the convex case where the definition of an equivalence relation is much easier. All of the above possibilities except R4 collapse into one. It is adequate to say that two points in the viewing space of a convex polyhedron are equivalent if and only if the same faces of the polyhedron are visible; this ensures that equivalent points have topologically equivalent images and enforces a continuous path requirement. The resulting equivalence classes of the viewing space are called principal view regions. Loose reference is made to the images of points in a particular principal view region as the principal view of that region. The set of faces visible from a particular view region is called the *face set* of that region.

5. The Two-Dimensional Problem

It is instructive to consider the problem in two dimensions. Polyhedra become polygons, perspective viewing space becomes the plane, and orthographic viewing space the unit circle. An image is a line segment subdivided into segments which are the images of distinct edges. One can imagine a two-dimensional viewing bug crawling around in the plane or on a circle surrounding a polygon. Even in the non-convex case, an equivalence relation on the points of the viewing space is not hard to define. An edge of a convex polygon can be entirely visible in the image or completely invisible. In addition, for the non-convex case, an edge may be partially visible in one of three ways: either end point or both may be obscured. Two images are equivalent if and only if the same edges and vertices of the polygon are visible. The requirement that the same set of vertices is visible distinguishes between being able to see part or all of an edge.

First consider convex polygons under perspective projection. The region of the plane from which a particular edge of a convex polygon is visible can be found in the following way. Simply extend the edge to a line (infinite in both directions). This line divides the plane into two regions. The face is visible exactly on the side of the line away from the polygon. Call this line an "edge line." For a convex polygon construct all of the edge lines. This partitions the plane into polygonal regions and each region has a different set of visible edges. These regions are the principal view regions. Unbounded regions correspond to the view regions possible under orthographic projection. Alternately, for orthographic projection, consider the directions from which a face is visible using the unit circle as the viewing space. The hypothetical bug crawling around a circle will begin to see an edge just when her viewing direction has gone past the direction of the edge. Thus for each edge, superimpose a line on the unit circle passing through the center and with the same direction as the polygon edge. Label the two intersections of line and circle appropriately for the direction in which the edge may be seen. After this is done for all edges, the unit circle is divided into arcs, each of which correspond to a distinct view of the polygon. Figure 3 illustrates these ideas for a pentagon.

6. An Algorithm for Convex Polygons

In this section, a formal algorithm is given for enumerating the principal views of a two-dimensional convex polyhedron under perspective projection. (The two-dimensional algorithm turns out to be useful in developing an algorithm for three dimensions.) It uses the "plane sweep" technique of computational geometry. (See Preparata and Shamros (1985), pages 10-11, for a description of plane sweep.) The idea is to sweep a line down through the plane, starting well above the polygon. At the beginning, output a description of the views from all the regions that this initial line intersects and set up a description of the intersection of the sweep line with the edge lines and viewing regions. Now sweep the line down through the plane. Each time a new region is encountered, give a description of the new view and update the sweep-line description. Continue until the last region vertex has been encountered

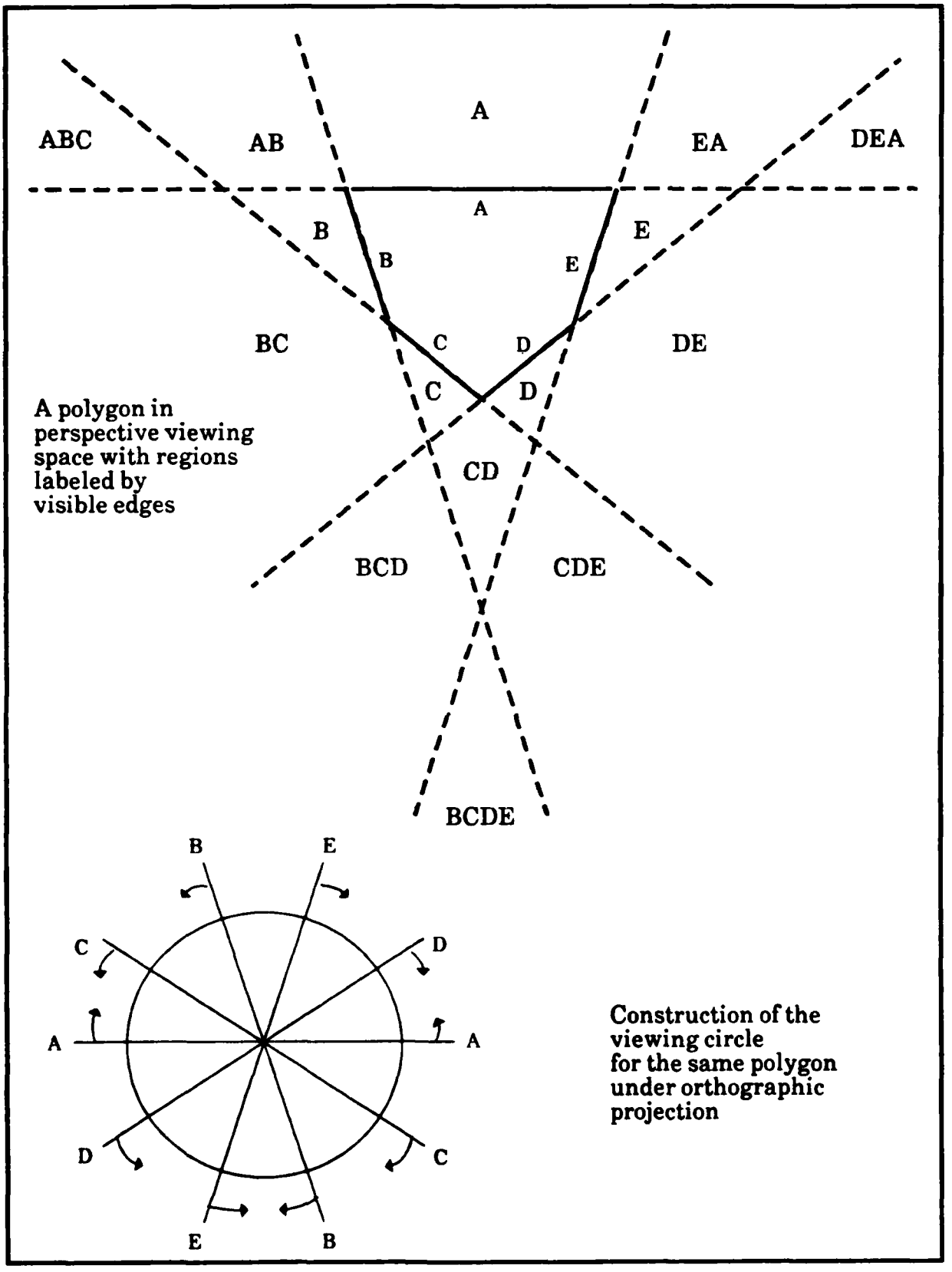

A
ABC
AB
EA
DEA
A
B
E
B
E
BC
C
D
DE
C
D
CD
BCD
CDE
BCDE
A polygon in perspective viewing space with regions labeled by visible edges
B
E
C
D
A
A
D
C
E
B
Construction of the viewing circle for the same polygon under orthographic projection

Figure 3

by the sweep line; at this point all views have been enumerated. Each view region is described by the set of edges visible from the region. Call these sets "principal edge sets." (In this two-dimensional convex case, the sets are almost enough to reconstruct each view since only an ordering must be added to the edge sets. The ordering corresponds to the ordering of edges around the polygon.)

More precisely, let a convex polygon in the plane be given. Extend each edge to an edge line. Calculate all intersections between distinct edge lines; suppose these points are $v_1, v_2, ..., v_m$. For convenience assume no edge is vertical and that no two v_i are on the same horizontal line. (If not, rotate the polygon a bit.) Note that no three lines can intersect at a point. Determine whether the polygon is to the right or left of each edge. Sort the intersection points by y coordinate. Renumber according to this sorting so that v_1 has the greatest y coordinate, v_n the smallest. Choose y_0 greater than the y coordinate of v_1. Consider the line L given by $y = y_0$. It is above all the intersection points and will be the initial sweep line. The event point schedule (list of points where changes must be made) is $v_1, v_2, ..., v_m$. Calculate the intersections of all the extended edges with L, finding the x coordinate of those intersections. The sweep-line status structure (description of the sweep line) is a list of the edge lines ordered from left to right according to the x coordinates of their intersection with the sweep line. Sort the x coordinates of the intersections, remembering to which line they correspond, and get the initial sweep-line status structure. Construct the set of edges visible from each of the $n+1$ principal view regions that L intersects as follows. Begin with $n+1$ ordered empty sets, one for each region intersected by L. For each of the lines K intersecting L, place the edge associated with K in all of the sets for the regions to the right or left of the K's intersection with L depending on whether the polygon is to the right or left of K. When all of the lines have been processed, these sets will contain the appropriate elements. Output them.

Now sweep the line down through the plane. An event occurs when the sweep line encounters an intersection point v_i. After the sweep line passes v_i, it will have one view region removed from the set of regions it intersects, and one added. For each v_i proceed as follows. Suppose v_i is the intersection of lines K and L. K and L will be adjacent to each other in the sweep line status structure. Reverse them. Output an edge set for the new region encountered (except in the one instance that the new region is the polygon itself). The edge set may be calculated by taking the edge set of the region the sweep line is leaving and simply reversing the answers to the questions "is K visible?" and "is L visible?" This is true since all other edges will have the same visibility in the new region. Continue until all the intersection points have been crossed.

7. The Complexity of the Algorithm and the Number of Principal Views of Convex Polygons

A formula is given for a much more general situation by Edelsbrunner, O'Rourke, and Seidel (1983) that can be used to determine the number of principal views of a

convex polygon. The details of application of the formula are omitted. If no two edges are parallel, the edge lines partition the plane into $C(n,2) + C(n,1) + 1$ regions. One of these regions is the polygon itself, so there are $C(n,2) + C(n,1)$ or $(n^2+n)/2$ regions. Every pair of parallel edges simply reduces the number of intersections, and hence principal views, by one. Thus the number of principal views is $(n^2+n)/2 - m$ where n is the number of edges in the polygon and m the number of pairs of parallel edges. (This formula also appears in Werman, Baugher, and Gualteri (1986).)

What is the complexity of the algorithm? Briefly, since each edge set may contain about half of the n edges, the size of the output is $O(n^3)$. It is not hard to see that the complexity of the algorithm is driven by this bound--all other parts, even sorting $O(n^2)$ numbers, are of lesser complexity.

Calculating orthographic views in two dimensions for convex polygons is very simple since we really need only find the slopes of the edges and the direction of the polygon from each edge, sort the slopes and use this information to "walk" around the viewing-space circle, outputting principal edge sets. Suppose the polygon again has n edges. If there are no parallel edges, the viewing circle will be divided into $2n$ parts and there will be $2n$ principal view regions.

8. Non-Convex Polygons

What if the polygon is not convex? First consider Figure 4, where the simplest possible non-convex polygon is shown. As in the convex case, lines are drawn that partition the plane into distinct view regions. However, some of these are not complete lines but instead are rays. Another difference is that there are regions where only part of an edge is visible. These are the main differences between the convex and non-convex case. Figure 5 illustrates another basic concept. Some edges of a polygon will be on the convex hull, some not. Those which are not on the hull occur in adjacent groups of at least two such edges, with a convex hull edge on either side. Call such a group a "gouge." On the figure, there are three gouges, labeled G1, G2, and G3. Consider the polygon which is the convex hull of the original polygon. Each gouge has one edge of the convex hull polygon associated with it. The other edges of the new polygon are edges of the original polygon. The principal view regions of the new polygon can be constructed. In effect, these regions correspond to the equivalence relation given by: two points are equivalent if and only if the same convex hull edges of the original polygon are visible and some part of the same gouges are visible. When envisioning the picture, it is helpful to think of these regions giving the "big" picture; when the details of the gouges are taken into account, the resulting principal view regions will be subsets of the convex hull view regions. Moreover, to get these details, we can consider one gouge at a time without worrying about interactions with the edges in other gouges.

In Figures 6a and 6b we see gouges with three edges. The rest of the polygon has been depicted by wavy lines; except for the assumption that the end points of the gouge are on the convex hull (because they have just three edges and must have adjacent convex hull edges on either side) the rest of the polygon is not presently of

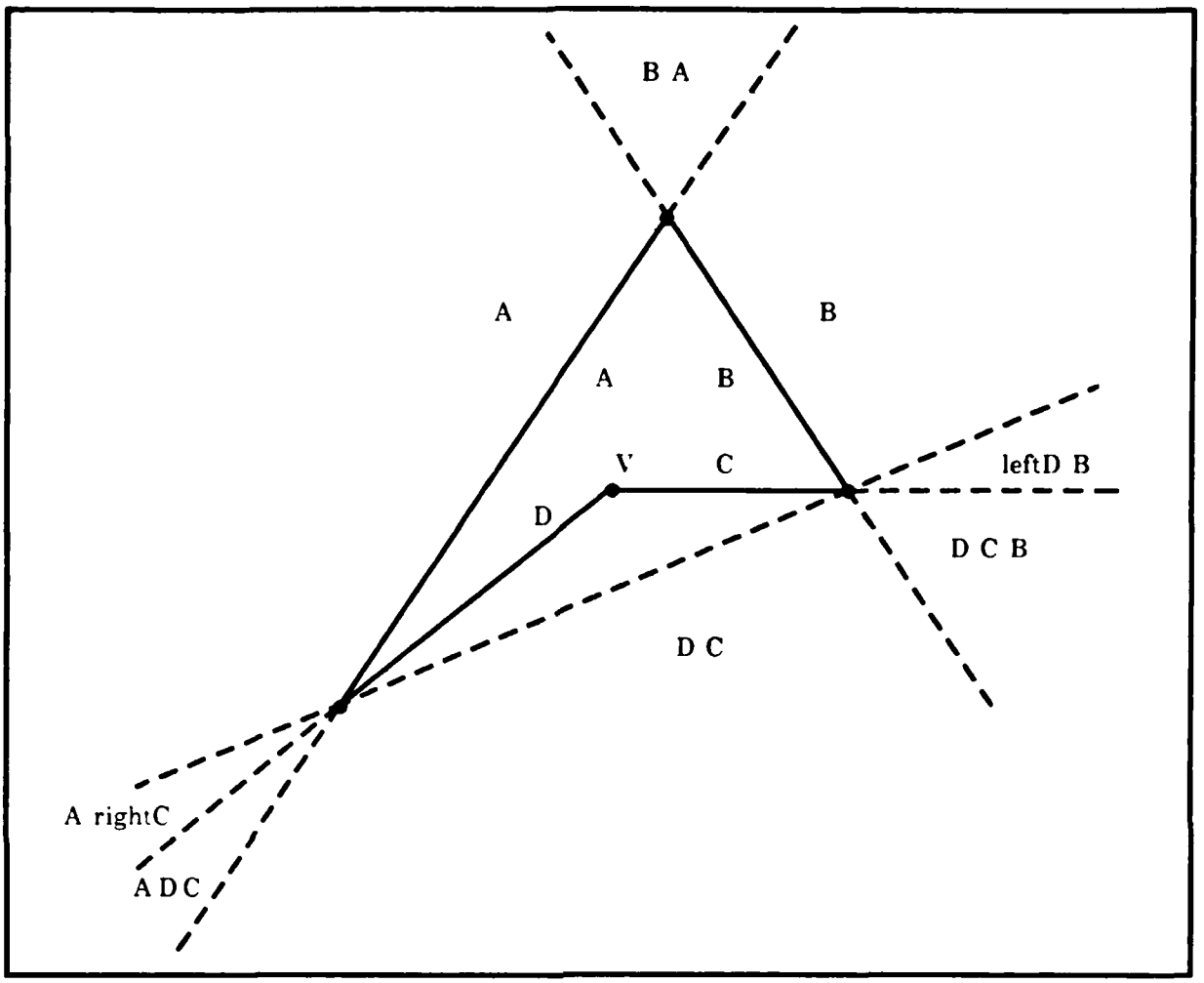


Figure 4

interest. The two figures are almost the same, the only difference being whether or not the rays r1 and r2 intersect. Regions are labeled by the edges of the gouge which are visible. In the second case, there is a region in which only the middle of edge B is visible.

Next consider the general case of a vertex inside of a gouge and not an end point. Change the point of view for a moment and think of a bug sitting at that vertex and staring out at the world. If she can see any of the world outside of the convex hull, she will see it framed to her right and left by pieces of the polygon. In fact, she will see an infinite sector of the plane, as shown in the two examples of Figure 7 for a vertex v. The pieces of the polygon that frame the view will be other vertices of the gouge. Call this sector the viewing sector for the vertex. Call the rays that bound the viewing sector the right and left view bounds (where the meaning of right and left is taken from a point of view at the vertex). A vertex has a non-empty viewing sector just in case it is visible from outside the convex hull.

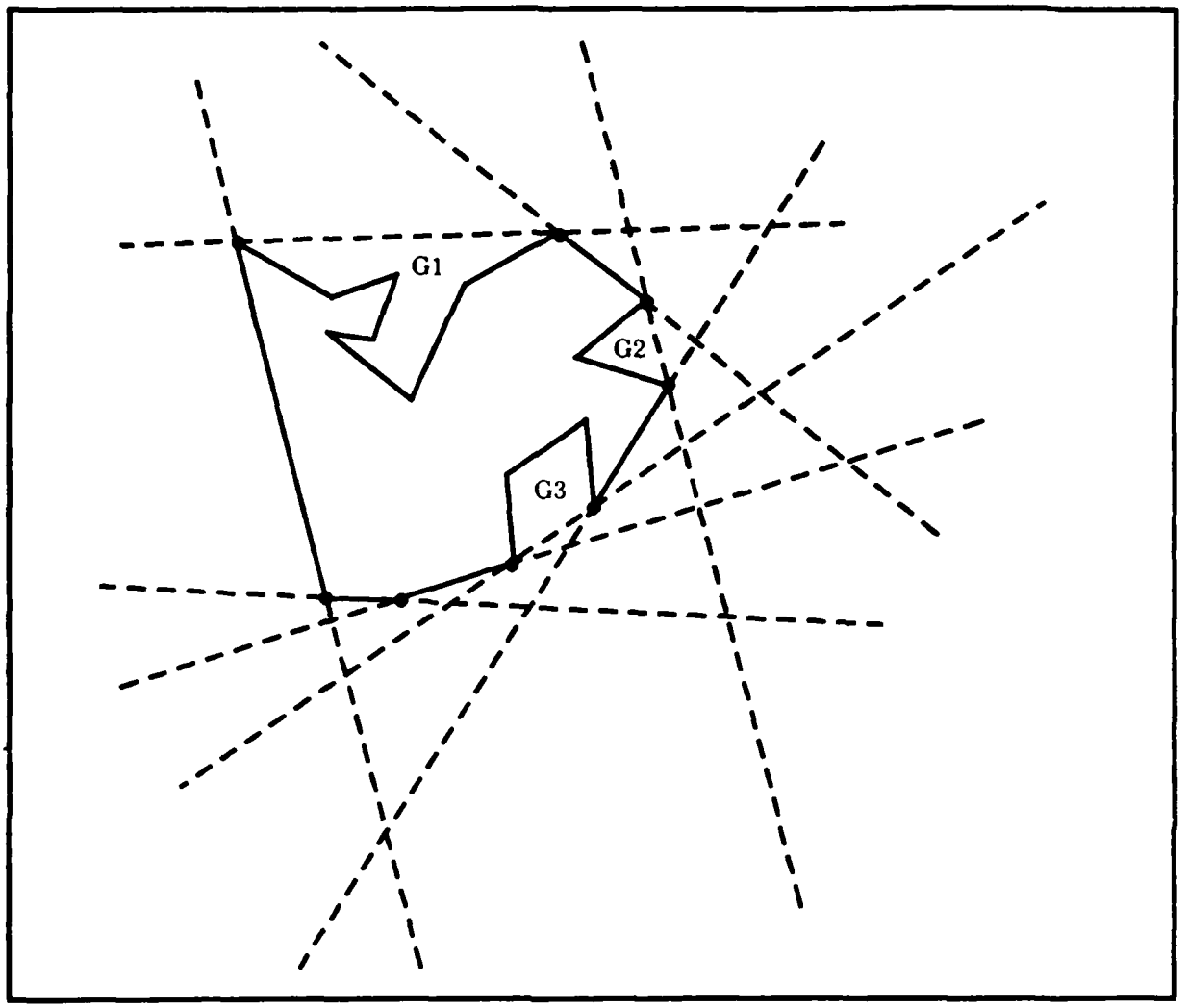


Figure 5

It is convenient to distinguish between the edges within a gouge that have one end point on the convex hull and those that do not. Define the first kind of edge to be an outer edge of the gouge and the second an inner edge. Now consider an inner edge within a gouge. If it is visible in its entirety from somewhere outside the gouge, the end points are visible so they each have a viewing sector. The two ways these can interact for an edge E are shown in Figure 8a. An indication is given of where and how the edge is visible. Figure 8b shows a couple of examples. Figure 9 shows examples of the two possible cases of an outer edge E in a gouge; the outer edge is either on the right or left of the gouge. An outer edge is always completely visible in some place.

There are several possibilites for edges that are not entirely visible from any place outside the convex hull. Some edges are simply not visible at all. An example

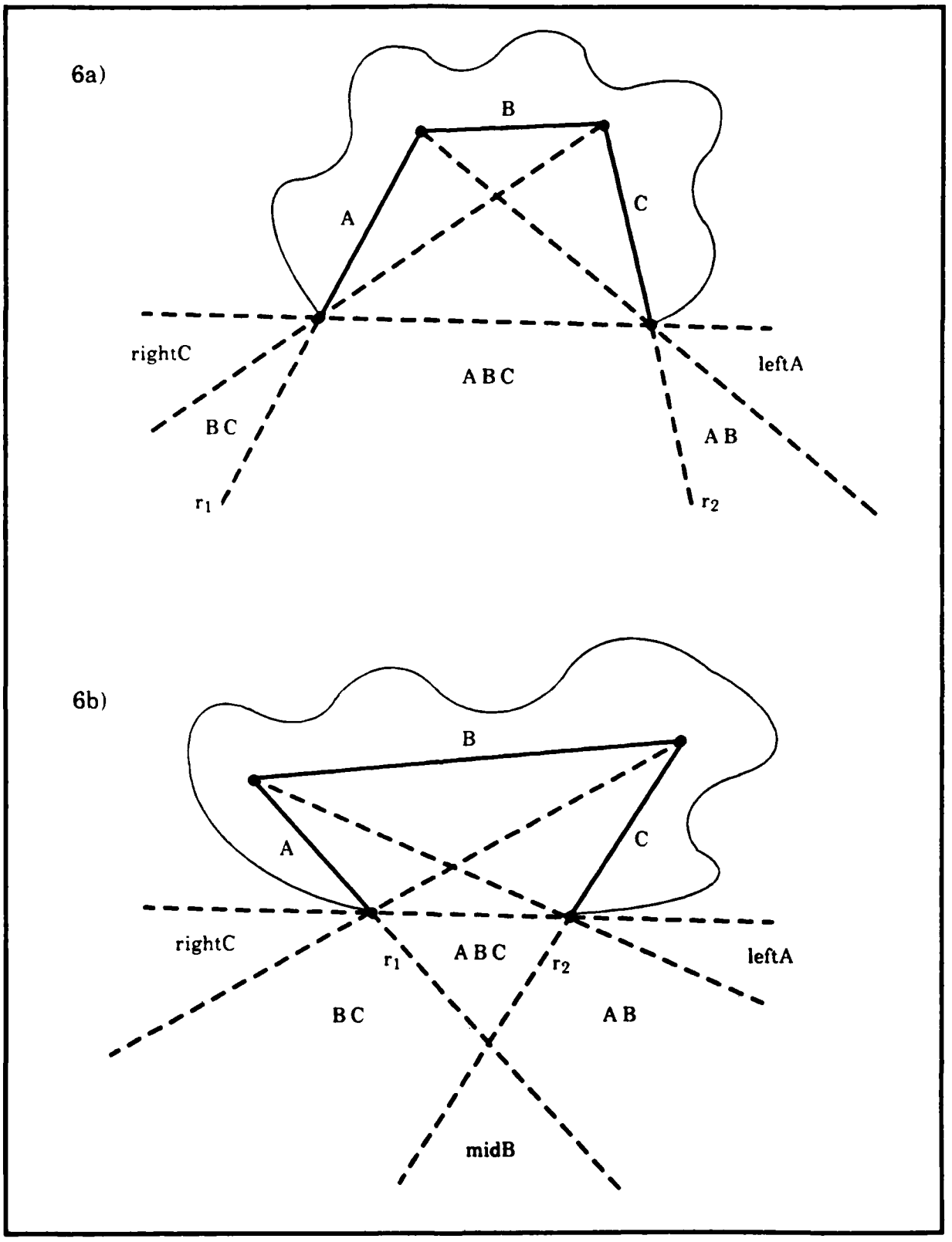

6a)
B
A
C
rightC
ABC
leftA
BC
AB
r1
r2
6b)
B
A
C
rightC
r1
ABC
r2
leftA
BC
AB
midB

Figure 6

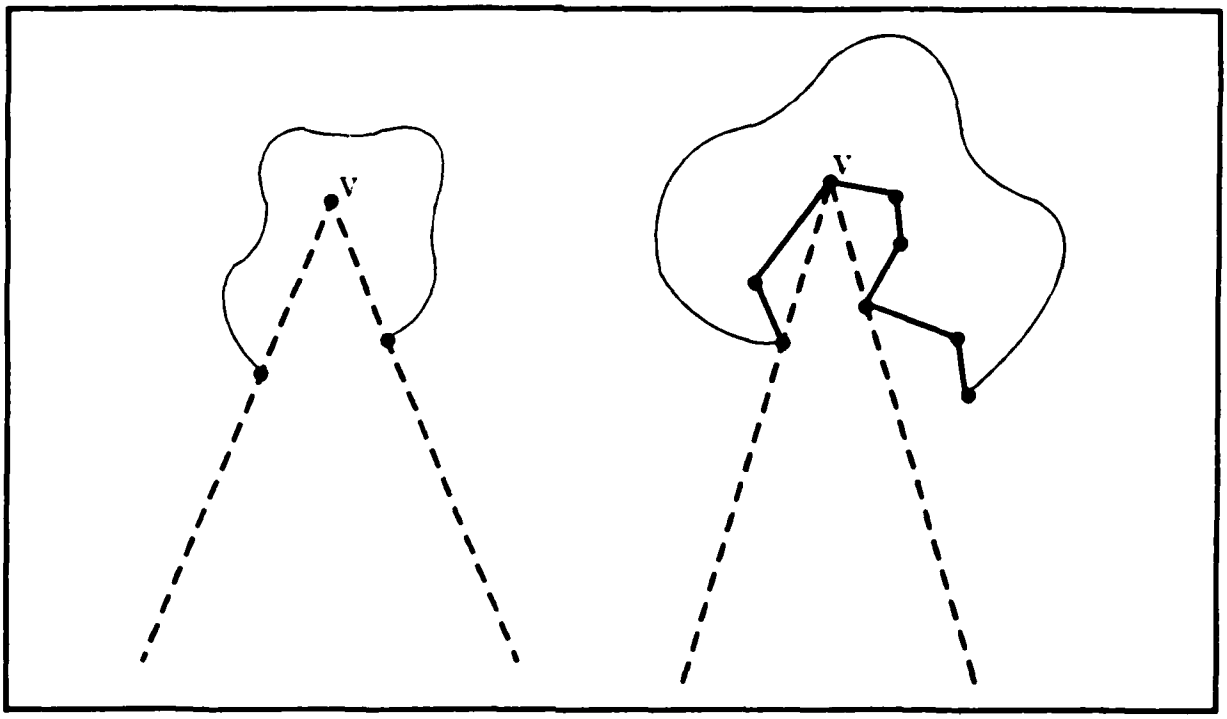


Figure 7

of the most confusing situation is sketched in Figure 10. Here the end points of the edge *E* are never visible, yet the middle of the edge can be seen. In some sense the middle of the edge has a viewing sector, and it is framed on the right and left along lines of sight passing from the edge to a vertex on one side of the edge to a vertex on the other side out into an infinite ray. For edges, call these rays the right and left view bounds, as for vertices. Not shown, but also possible, are cases in which one end of an edge but not the other is visible. Here, there will be one edge view bound.

These ideas can be put together to create a method for drawing the lines and rays that define the view regions of a non-convex polygon; the lines and rays can also be given labels to indicate all changes in the visiblity of edges when the edge is crossed. (The labels must be more complicated than in the convex case where it is simply the case that one edge is visible on one side of an edge line and invisible on the other. For example, crossing a ray might cause one complete edge and also one side of another to disappear.) An outline of the method follows. To simplify the description, it will be assumed that within a gouge there are no three co-linear vertices.

First construct the convex hull of the polygon. Extend all of the edges of this polygon into full edge lines, some of which are not true edge lines but lines associated with gouges. Label the true edge lines to indicate on which side the associated edges can be seen. For each gouge do the following:

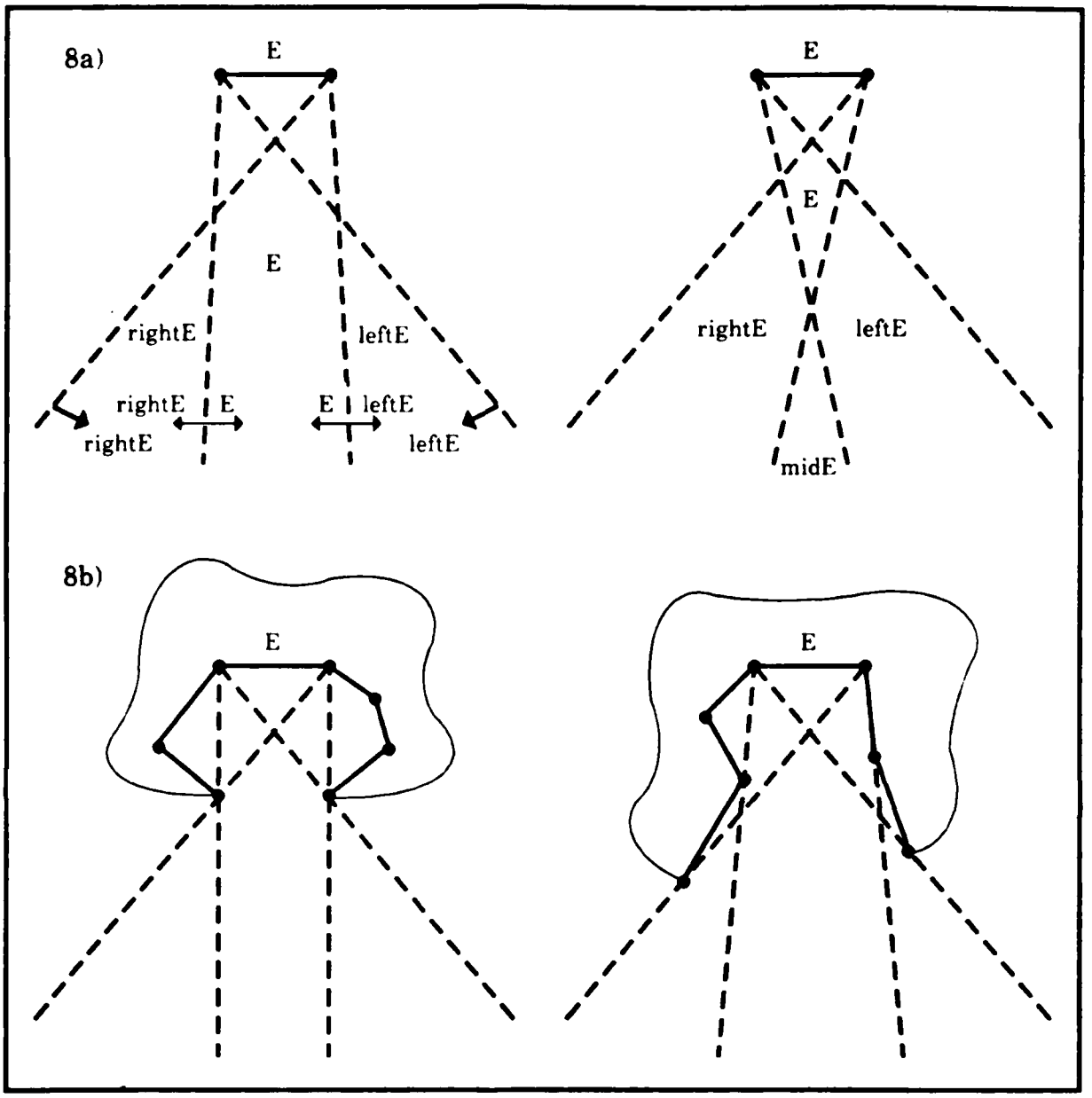


Figure 8: (a) Basic Picture; (b) Examples

Let $V_1, V_2, \ldots V_m$ be the points on the gouge ordered in clockwise fashion as in Figure 11. For each V_i, $1 \leq i \leq m\text{-}1$, proceed as follows. For the purpose of measuring angles, pick a point Q in the interior of the polygon such that the line segment from V_i to Q is in the interior of the polygon. For each j, $1 \leq j \leq m$ and $j \neq i$, let θ_j be the angle $Q\ V_i\ V_j$ measured in the clockwise direction. Find r with $1 \leq r < i$ such that $\theta_r = \min \{\theta_j \mid 1 \leq j < i\}$ and l with $i < l \leq m$ such that $\theta_l = \max \{\theta_j \mid i < j \leq m\}$. V_i is visible from outside the polygon just in case $\theta_r > \theta_l$. If it is visible, the rays from V_i to V_r and beyond and from V_i to V_l form the view bounds of

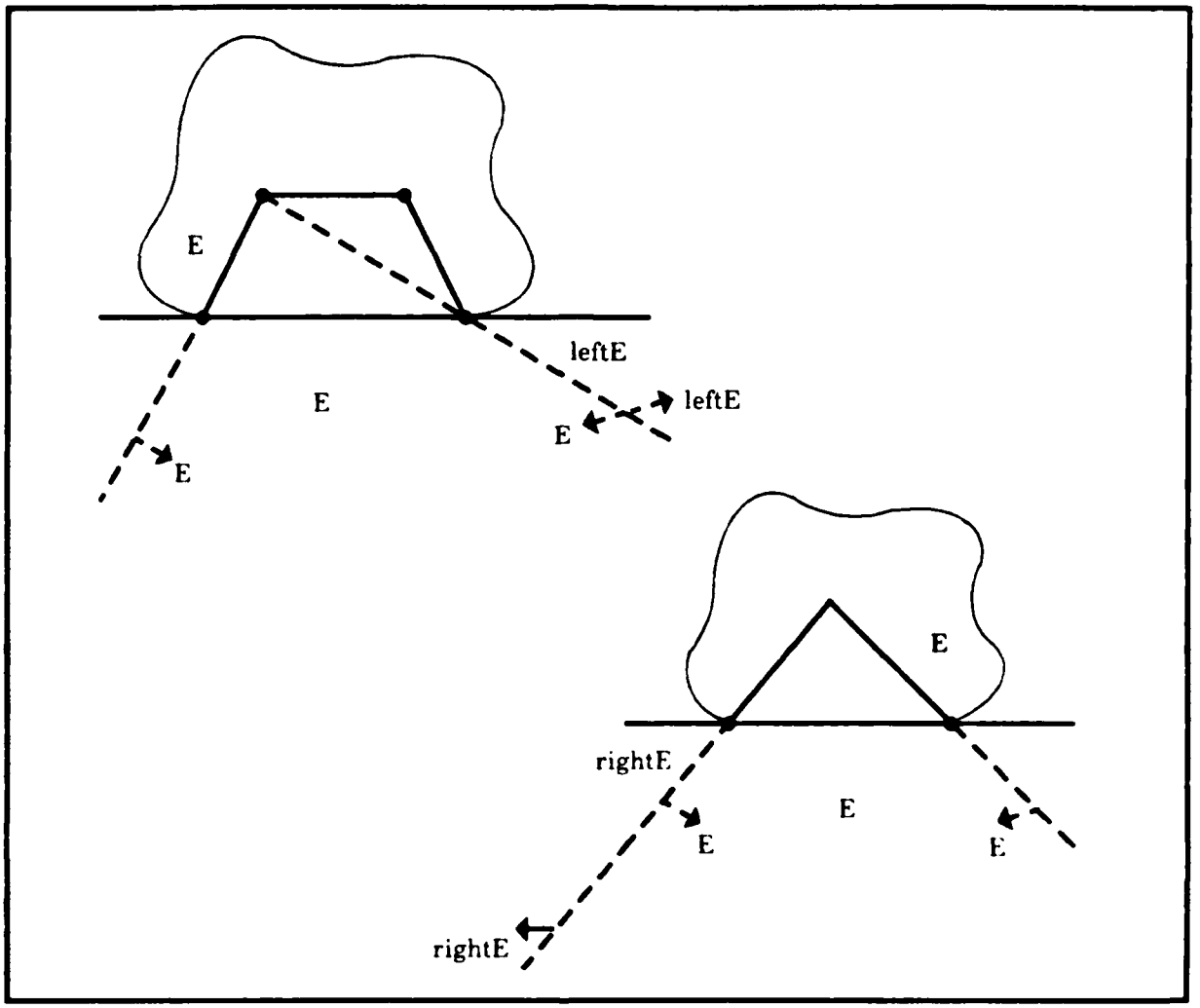


Figure 9

its viewing sector. If V_i is visible, draw these rays. Since the view bounds for V_i may possibly also be view bounds for edges, they may need to be extended. If the ray V_i to V_r can be extended backwards without intersecting the polygon interior, do so as far as possible. In this case, the end point of the ray is now a point on an edge e of the polyhedron. Record that fact in a list of possible right view bounds for e. Treat the other view bound similarly. Figure 11 illustrates an example of this construction with $m = 7$ and $i = 3$.

Once the vertices of the gouge are all processed, consider the edges of the gouge. For each edge e, if both vertices are visible, nothing more need be done. If neither vertex is visible, look to see if possible view bounds are recorded for e. If none are, e is not visible. Otherwise pick the possible right and left bounds that are farthest apart; these are the view bounds for e. In a similar way, find the one view bound for e if just one end point of e is visible.

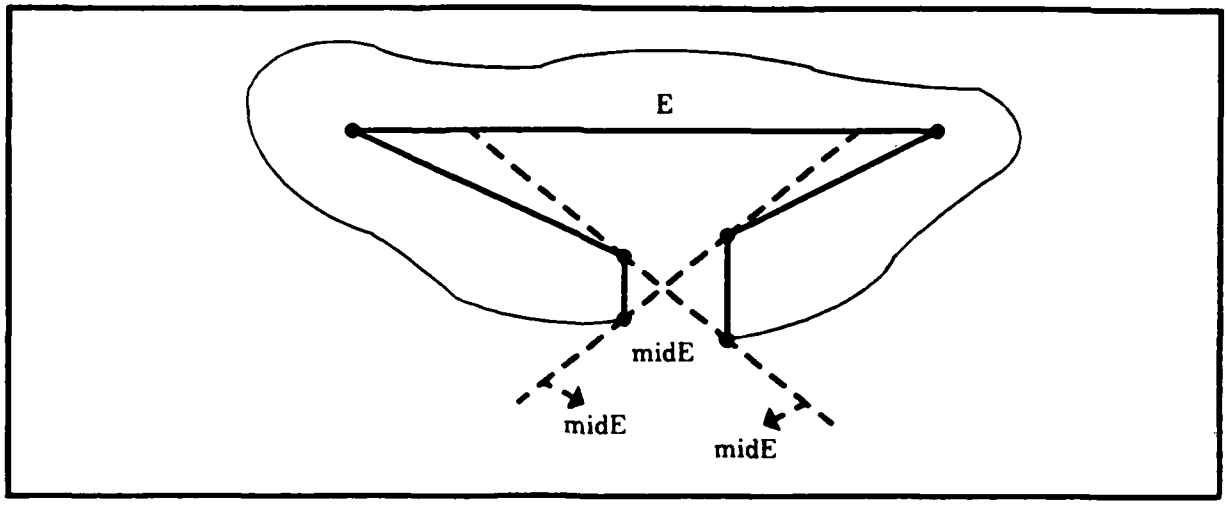


Figure 10

Of course, all these rays must be labeled. The details are left to the reader. It is not hard to see that for a gouge with m vertices, the construction for the gouge itself requires $O(m^2)$ time.

This gives a sketch of how to construct the lines and rays in the viewing plane for the non-convex case. Once this is done, an algorithm similar to the one given for the convex case could be implemented. The data structures required would be more complex because of the presence of rays, as well as full lines, and the more complicated labels.

9. An Algorithm for Finding the Principal Views of a Convex Polyhedron

The ideas in the algorithm for the two-dimensional convex case can be used to create an algorithm for three dimensions, essentially by everywhere adding a dimension. The new algorithm will produce a listing of the visible faces for each principal view region of a convex polyhedron under perspective projection. As in the two-dimensional case, this listing, along with a knowledge of the geometry of the polyhedron, is enough to specify the geometry of the views.

For each face, extend the face to a full plane and call this the face plane for the face. The face is visible exactly on one side of its face plane, the side away from the polyhedron. Thus, by constructing all the face planes, we divide three-dimensional space into regions, each of which has a distinct principal view. Instead of a sweep line, a sweep plane will be used. See Preparata and Shamros (1985), pages 299 and 313.

A sweep-plane status, i.e., description, is required, which describes the intersection of the sweep plane and all the face planes as the sweep plane sweeps through space. The intersections are lines in the sweep plane; we also need to know

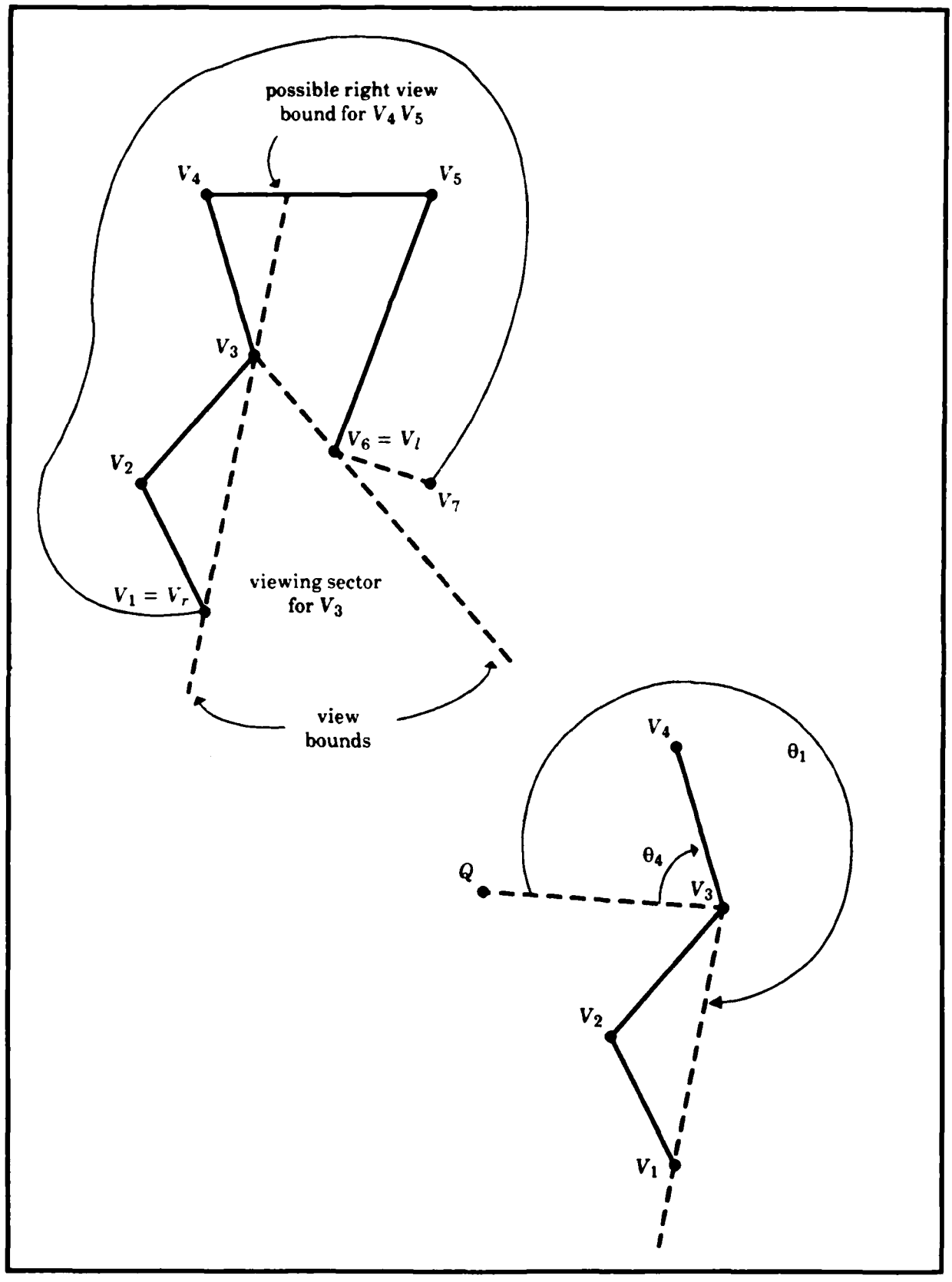

possible right view
bound for $V_4 V_5$
V_4
V_5
V_3
$V_6 = V_l$
V_2
V_7
$V_1 = V_r$
viewing sector
for V_3
view
bounds
V_4
θ_1
θ_4
Q
V_3
V_2
V_1

Figure 11

the side of each line on which the corresponding face is visible. Thus the needed information can be obtained from a description that is similar to Figure 3. It is not necessary to know the coordinates of the intersection of each pair of lines, but we do need to know their relationships. One sufficient representation is this. For each line, choose an arbitrary direction along the line. Give three ordered lists for the line: one containing lines; the other two, regions. Traveling along the line in the given direction, the line list should give the other intersecting lines as encountered; one region list is a list of regions found to the right of the line, the other regions to the left. Assume that no face is horizontal (otherwise change the coordinate system a bit) and, for the moment, that no four face planes intersect at a point.

Initially, calculate the intersections of all triples of planes that have an intersection. Call these points $v_1, v_2, \ldots, v_m$. Order the points by z coordinate, and rename them according to the ordering with v_1 being the greatest. Choose z_0 to be greater than the z coordinate of v_1. Let S be the plane given by $z = z_0$. Assuming the z-axis is vertical, all the v_i are below S and it will be the initial sweep plane. Now the initial sweep-plane status must be calculated and the face sets for all the regions S intersects must be output. Finding the face sets corresponds almost exactly to solving the two-dimensional problem of finding edge sets and a method for this has already been given. (Of course the output does not omit a region corresponding to the interior of the polygon.) The algorithm for the two-dimensional case can easily be modified to construct an initial sweep-plane status, and the details are left to the reader.

As the sweep plane starts sweeping down through space, distances will change but the basic relationships of lines to each other will change only when one of the v_i is encountered. What happens then? Just as in 2-space, the intersection point will be the lowest vertex of some viewing region that the sweep plane is about to stop intersecting and the highest vertex of a new region that it is about to intersect. Figure 12 illustrates an example of how the sweep plane changes. Only the data for the three face planes, A, B, and C, that form the intersection are shown. The arrows on the lines indicate where the faces are visible. One region, labeled AB'C' in the figure (meaning that A is visible but B and C are not), disappears in the "after" figure and a new one, labeled A'BC, appears. The lines shown in the diagram all must have data modified in the sweep-plane status structure, but it is important to notice that no other changes need be made any place in the sweep plane. The visibility of faces other than A, B, and C will not change in any region. Other faces will be visible in the region labeled A'BC just in case they were visible in the region labeled AB'C'. Thus, the intersection points can easily be processed in their sorted order; simple calculations can be used to update the sweep plane and output a new viewing region for each intersection. When all intersection points have been processed, all the regions will have been found.

An assumption is often made in the study of polyhedral worlds that the vertices of all polyhedra are trihedral, that is, every vertex is at the intersection of exactly three faces. The restrictive assumption made above that no four (or more) planes intersect

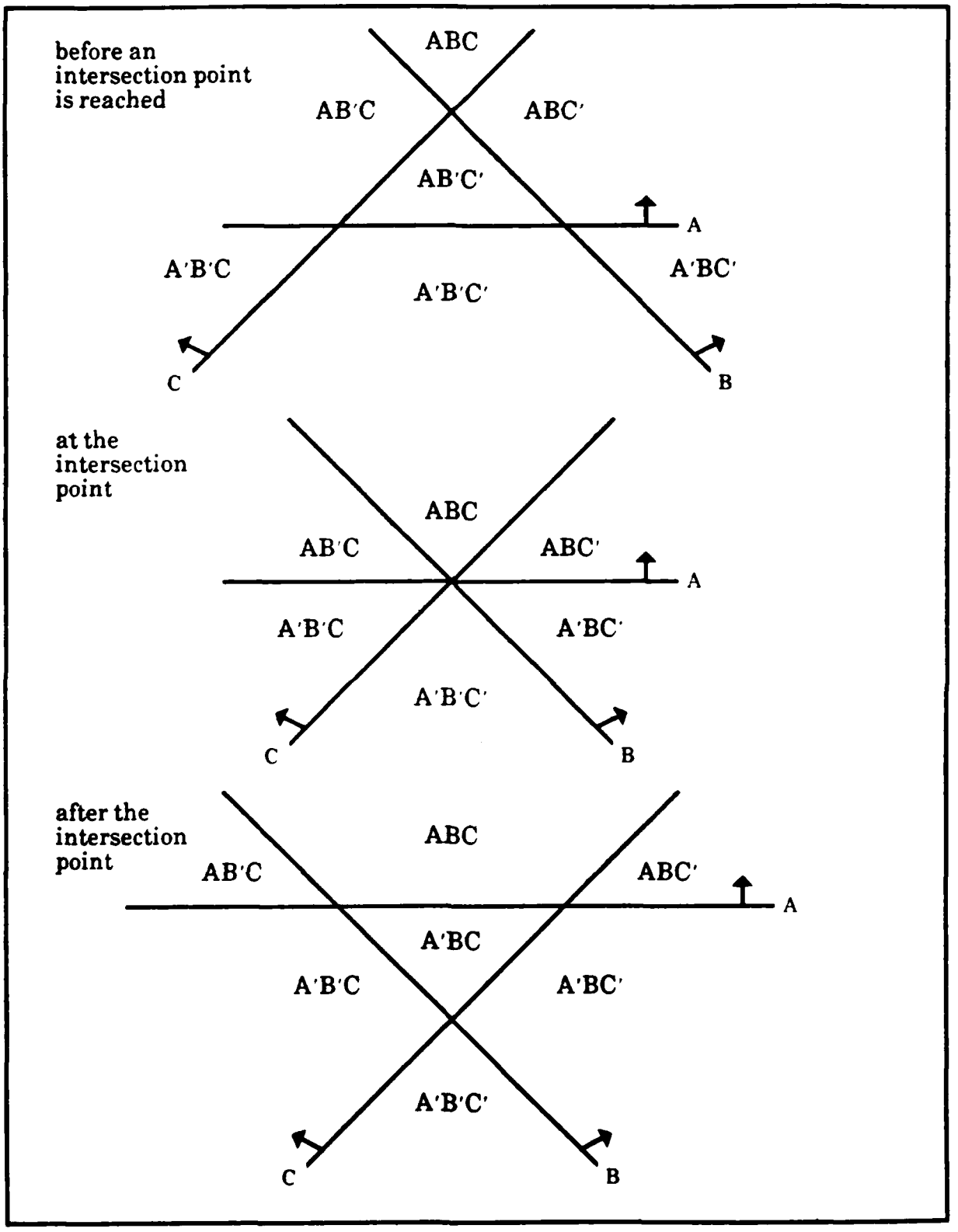


Figure 12: The Changing Sweep Plane

at a point is a sort of generalized trihedral polyhedral world assumption since it was made for all intersections of face planes, not just those that happen to be vertices of the polyhedron. With some trouble, this restriction can be avoided. Consider first what happens when exactly four planes meet at a point v_i. Figure 13 illustrates an example of what a part of the sweep plane looks like when faces with intersection lines labeled L1 through L4 are about to meet at an intersection point v_i and after the last intersection point, $v_{(i-1)}$ has been encountered. As the sweep plane nears v_i, the quadrilateral in the figure and the two triangles will become smaller and smaller, converging to a point when the plane actually meets v_i. Shortly afterwards, the configuration must look like the last one shown in Figure 13. To see that this is so, consider what would happen if the planes were moved very slightly so that they didn't quite intersect. As the sweep plane moves downward it might first encounter, say, the intersection of the planes corresponding to L1, L2, and L4. The reader is left to trace through the configurations as triples of lines (L1, L2, L4), (L1, L2, L3), (L2, L3, L4), and (L1, L3, L4) one by one flip over each other, corresponding to intersections of triples of the corresponding planes and resulting in the last configuration of Figure 13. No other lines in the sweep plane can be involved in these configurations in the sense that, for each of the four lines, intersections with the other three must be adjacent with no other intersections in between. Otherwise, the other line in such an intersection would get swept into the intersection point v_i, contrary to the assumption just four planes intersect at v_i.

What happens if more than four planes meet? Figure 14 illustrates an example of the relevant portion of a sweep plane before and after six planes meet. (When five or more planes meet there are not unique before-and-after pairs of configurations but one member of the pair can be calculated from the other.) One way of understanding what happens is to assume the planes are moved slightly, as in the paragraph above. An easier way is this. Suppose that the coordinate system is such that the intersection point is the origin. Let us look at the sweep plane just before and after the intersection, say at $z = z_0$ and $z = -z_0$. Clearly a point on a line in the first plane which is at (x, y, z_0) will correspond to the point $(-x, -y, -z_0)$. Thus the configuration in the first plane will be changed into a new configuration resulting from reflection of the first configuration twice, once across the x-axis and once across the y-axis. This helps in comprehending construction of the new configuration from the old.

The construction is probably best understood by means of an example; the transformation of Figure 14 will be used. Here there are six (in general m) planes intersecting with corresponding lines L1 through L6. Around the periphery of the configuration there are twelve ($2m$) regions, labeled R1 through R12. Call these regions the "outer" regions of the configuration. Regions S1 through S10 each have all their edges in the set of lines L1 through L6. Call them "inner" regions. Each of the inner regions are about to disappear and will be replaced by a new corresponding inner region. For each of the lines, we need only consider the local relations of the lines, the six (m) intersecting lines along the line and the six (m) regions on each side of the line. Otherwise the sweep plane remains unchanged. Consider what happens first to the line intersections along one of the lines, and then the regions on either

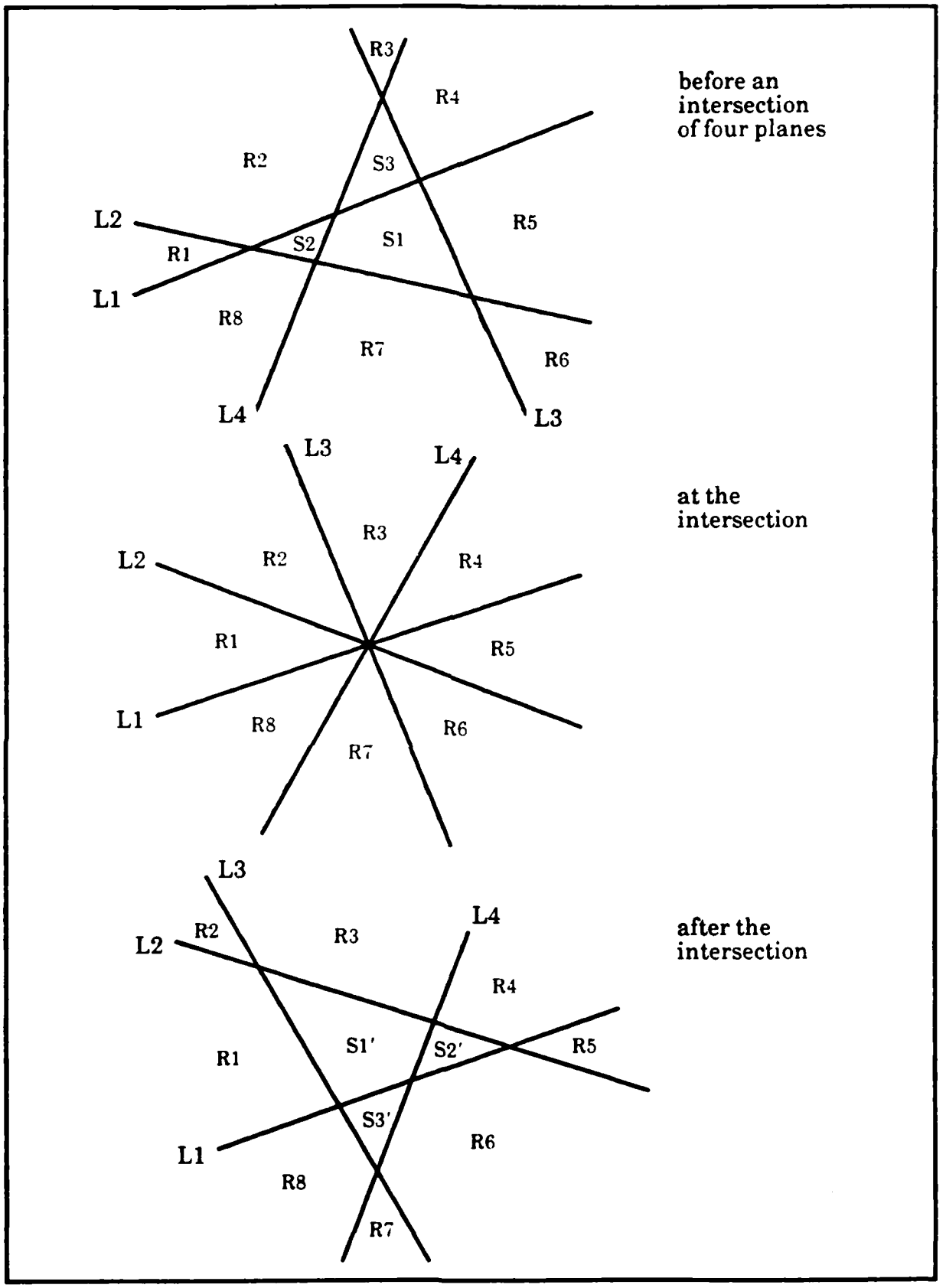

R3
R4
R2
S3
L2
R5
S2
S1
R1
L1
R8
R7
R6
L4
L3
before an intersection of four planes
L3
L4
R3
L2
R2
R4
R1
R5
L1
R8
R6
R7
at the intersection
L3
L4
R2
R3
L2
R4
R5
S1'
S2'
R1
S3'
R6
L1
R8
R7
after the intersection

Figure 13

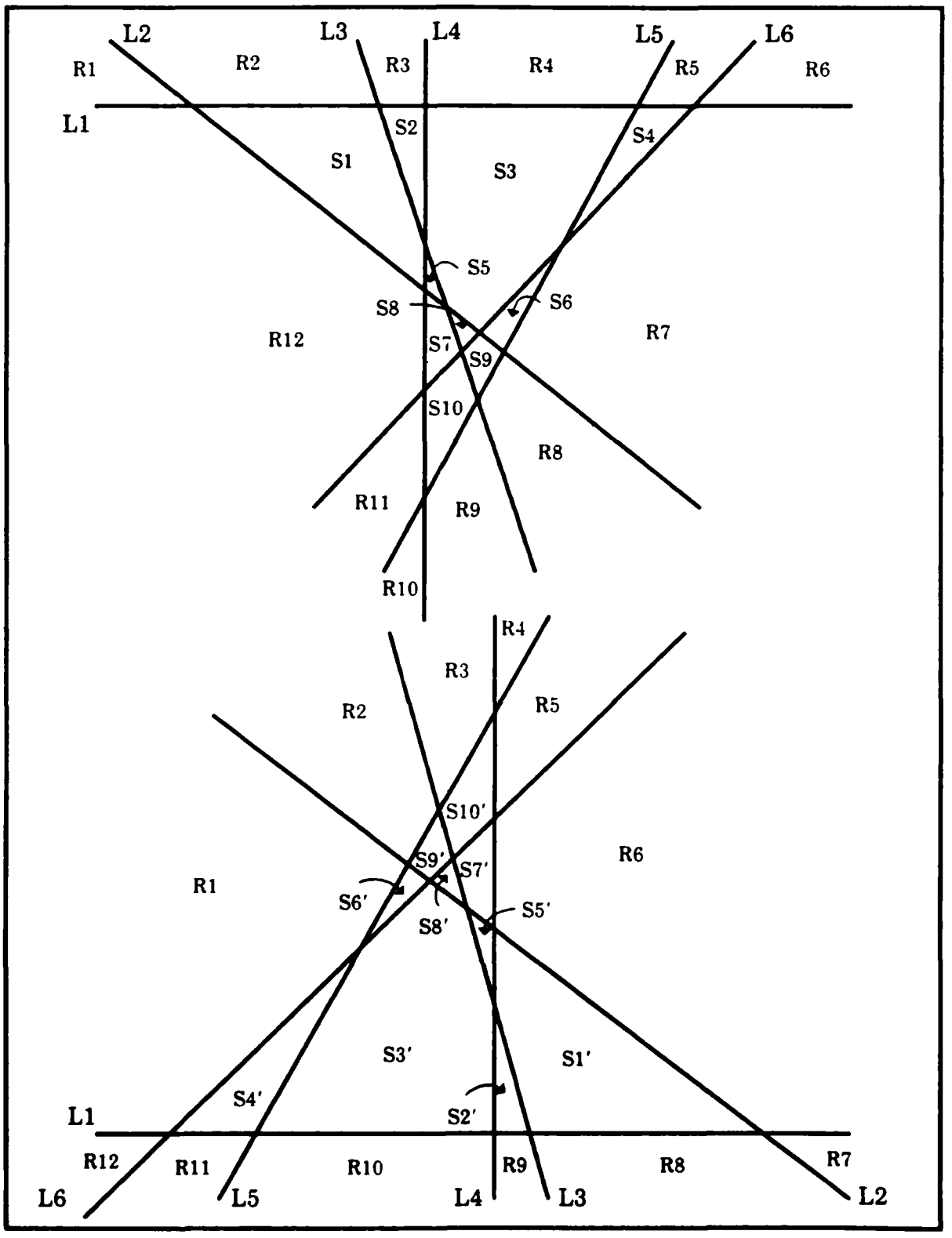

L2
L3
L4
L5
L6
L1
R1
R2
R3
R4
R5
R6
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
R7
R8
R9
R10
R11
R12
S1'
S2'
S3'
S4'
S5'
S6'
S7'
S8'
S9'
S10'

Figure 14

side. Due to the symmetry, the list of lines that intersect any given line is simply reversed in order. What happens to the regions is more complex. The four outer regions shown at the very ends of each of the lines do not change. All the others will, and the changes can be described in two steps. For each line, first, in the parts of the region-right and region-left lists between the other lines in the configuration, every region will be an inner or outer region. Replace inner regions by the corresponding new inner region. Replace any outer region by the new outer region that is opposite the old one on the periphery of the configuration. That is, replace R1 by R7, R2 by R8, etc. Now in the second step, take all of these changed regions and, in essence, perform the two reflections mentioned above. Reverse right and left, and place in the opposite order. These changes are illustrated for line L4 in Figure 15.

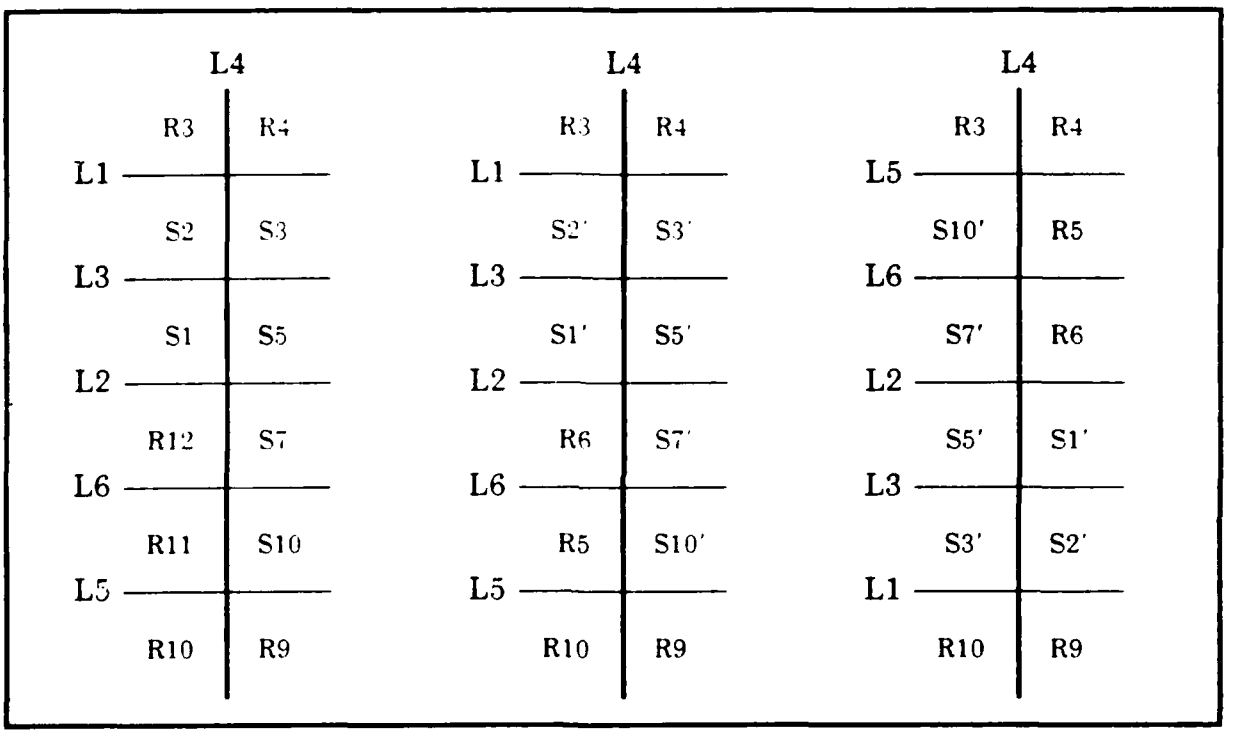


Figure 15

The algorithm given earlier for convex polyhedra with no intersections of four or more planes can be supplemented to deal with such intersections by recalculating the sweep-plane data structure in this more complex way when such a point is met, and by outputting the new regions which arise.

10. The Number of Views and the Complexity of the Algorithm for Polyhedra

How many principal views are there in the three-dimensional convex case? From the Edelsbrunner, O'Rourke, and Seidel paper cited above, an upper bound for the number of views is $C(n,3) + C(n,2) + C(n,1)$. The bound is actually achieved when every triple of face planes intersects and no four or more planes intersect at a single

point. The formula for this case can also be calculated by considering the number of view regions encountered by the sweep plane in the above algorithm. The initial sweep plane intersects $C(n,2) + C(n,1) + 1$ regions (from the two-dimensional case). There are $C(n,3)$ v_i, so $C(n,3) - 1$ view regions are encountered as the plane sweeps downward. (One intersection is encountered as the sweep plane is about to meet the polyhedron itself.) Hence the number of distinct principal views is $C(n,3) + C(n,2) + C(n,1)$. (This works out to be $(n^3 + 5n)/6$.)

If some triples of planes do not intersect at all, the number of points encountered by the sweep plane will be reduced and thus the number of principal views will be less. If m planes intersect at a point, where m is four or more, assuming no parallelism, just before the sweep plane meets the point, it can be shown that it will contain $(m-1)(m-2)/2$ inner regions that will disappear, to be replaced by the same number of new regions. Thus $(m-1)(m-2)/2$ new regions will be found by the algorithm in this situation in place of the $m(m-1)(m-2)/6$ that would be found if the face planes simply intersected in triples. Parallelism for this case reduces this amount even more.

It should be noted that, while parallelism and intersections of many face planes could occur, these are really unusual cases, except, perhaps, for parallelism in some man-made objects. Thus we have not only an upper bound, but a typical count of the number of regions.

As in the two-dimensional case, the time and space complexity of the algorithm is determined by the size of the output which is $O(n^4)$ (bits). The details are omitted.

For the orthographic case, suppose that no two edges are parallel. The part of the viewing sphere where a particular face can be seen is given by intersecting a plane with the viewing sphere to form a great circle. The face can be seen exactly on one side of the great circle. With no parallel edges all great circles are distinct, and each pair intersects at two points. Thus there are $2C(n,2)$ intersections. To count the number of regions on the viewing sphere created by these circles, sweep a plane down through the sphere, arranging to encounter the sphere first at one of the intersections. Just after the first encounter, four distinct viewing regions will be intersected. As the plane continues to sweep, a new principal viewing region will be discovered each time one of the intersection points is met, except for the very last intersection with the point exactly opposite the first point. Thus there are $4 + 2C(n,2) - 2 = n^2 - n + 2$ principal view regions.

11. Implementation of the Algorithm for Finding the Principal View Regions of a Convex Polyhedron

The author has written a program that implements the algorithm described above. The input is a file which contains coeffiecents of the equations for face planes. For convenience, the equations are constrained to be of the form:

$$ax + by + cy = 1$$

In addition, for each face, a Boolean value indicates whether the face can be seen from the origin. (Requiring the equations to be in this form in essence means that the coordinate system must be chosen so that the origin is not on any face. That restriction makes it possible to tell easily whether a face is visible from a point (x, y, z) using the boolean value and checking whether $ax + by + cz$ is negative or positive.)

The program consists of three modules. In the first, the intersections of all triples of planes are calculated, providing they exist. Since updating the sweep plane would be complicated if any two distinct points have the same z coordinate, the coordinate system is changed so that this doesn't happen. In addition, the points are sorted by z coordinate and data is collected on intersections of more than three planes, should there be any.

While no great attempt has been made to make the program efficient, some care was taken in solving the triples of equations that give the intersections of planes. The method used may itself be of some interest. A reader familiar with Gaussian elimination will recognize that, if a straightforward but thoughtless standard attack is used to solve equations 1, 2, and 3, then equations 1, 2, and 4, then equations 1, 2, and 5, etc., much effort will be repeated. The program avoids this duplication and even uses "half-pivoting" to reduce numerical error in the following way. First the equations are sorted according to the magnitude of the coefficient of x. After sorting, with the obvious notation, $|a_1| \geq |a_2| \geq |a_3| \geq$, etc. The overall strategy is to solve all triples involving the first equation using the coefficients of the first equation as the first row in the augmented matrix for the system, then all triples involving the second but not the first, using the second equation coefficients in the first row, and so on. The first time around, when the systems involve the first equation, all possible general calculations for such systems are done once and for all. In essence, the first equation is divided by a_1, and x is eliminated in local copies of the other equations. That is, in terms of matrix manipulations, all work on the first column of all matrices involving the first equation is done once and for all, one time per row. Now the same trick is repeated in the next column (while still dealing with the case that the first equation is part of the system). The local copies of the other (changed) equations are sorted by the magnitude of the coefficients of y, largest to smallest, and processing begins with the equation with the largest y coefficient used as the second equation in the system and solving all such systems, then the next largest y coefficient, and so on. After all sets of equations involving the first equation are solved, sets involving the second but not the first are solved in a similar fashion, and so on.

In the second module, the initial sweep plane is found. Essentially, the algorithm given earlier for finding the viewing regions of a convex polygon in the plane is used, except that there is no real polygon. In addition, the initial sweep-plane data structure is constructed. The sweep plane always contains a collection of lines that are formed by the intersection of the sweep plane and the face planes. (Because of

the possible change of coordinates, no face plane is allowed to be horizontal; hence all face planes intersect the sweep plane.) The sweep-plane data structure consists of three lists for each of the intersection lines as described before.

In the third module, the sweep plane sweeps through space. Each time a new intersection point is encountered, a new viewing region is output and the sweep-plane data structure is updated. When an intersection of more than three planes is encountered, a more complex procedure is followed that outputs the perhaps many new view regions and updates the sweep plane.

The sweep-plane technique appears to be powerful for this type of computation. The program also, incidentally, finds the set of vertices for each region (and the polyhedron itself), calculates a point within each region, and determines whether the view involved could be obtained under orthographic projection. It would be easy to add other computations; relevant data could be calculated as the sweep plane sweeps through space.

The program itself was intended to serve as a basis or a part of other programs. For example, it has been used as part of a graphics program that prints line drawings of principal views of a polyhedron. Some examples of the output of the graphics program appear in Figures 16 through 18.

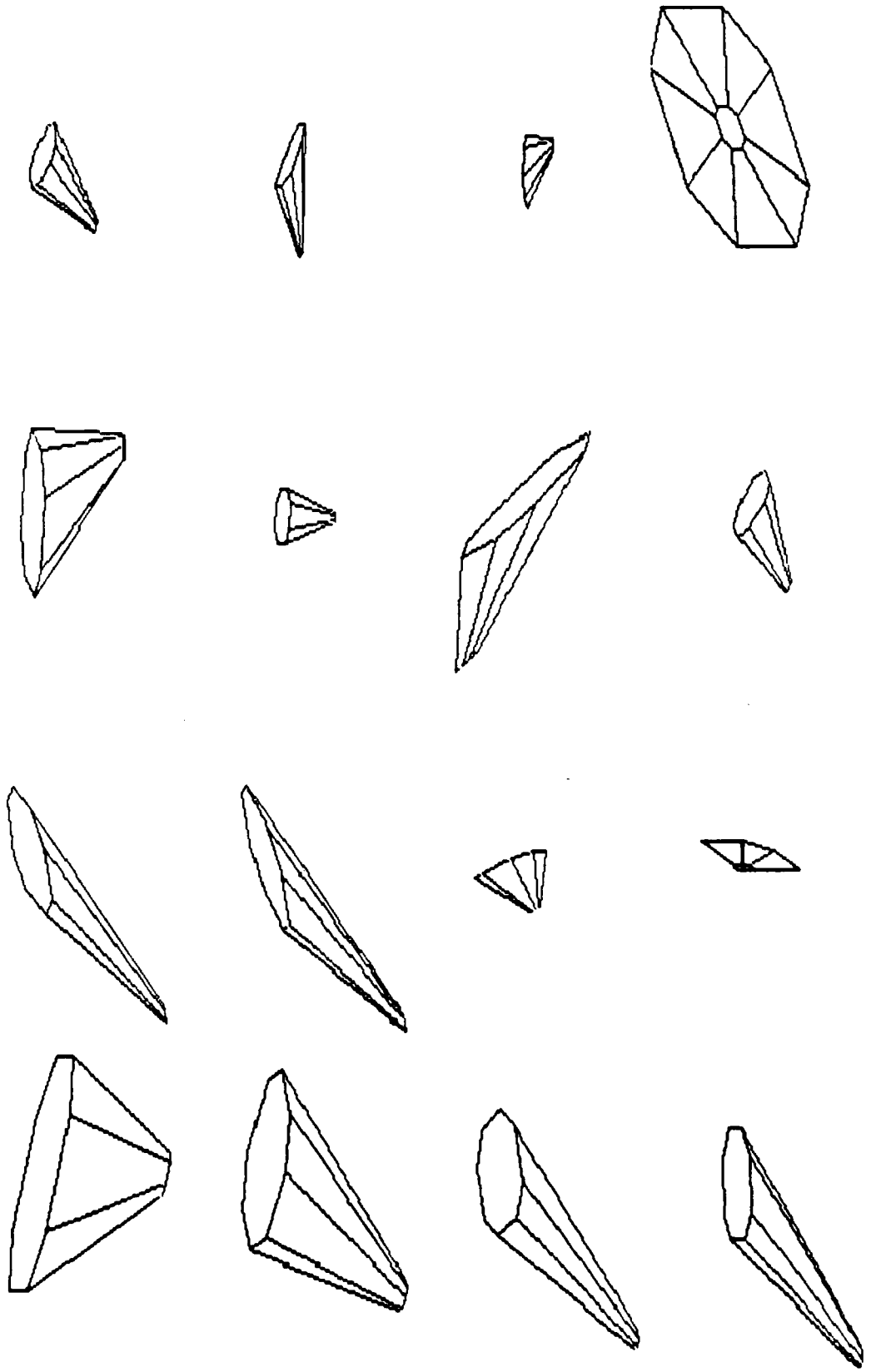


Figure 16

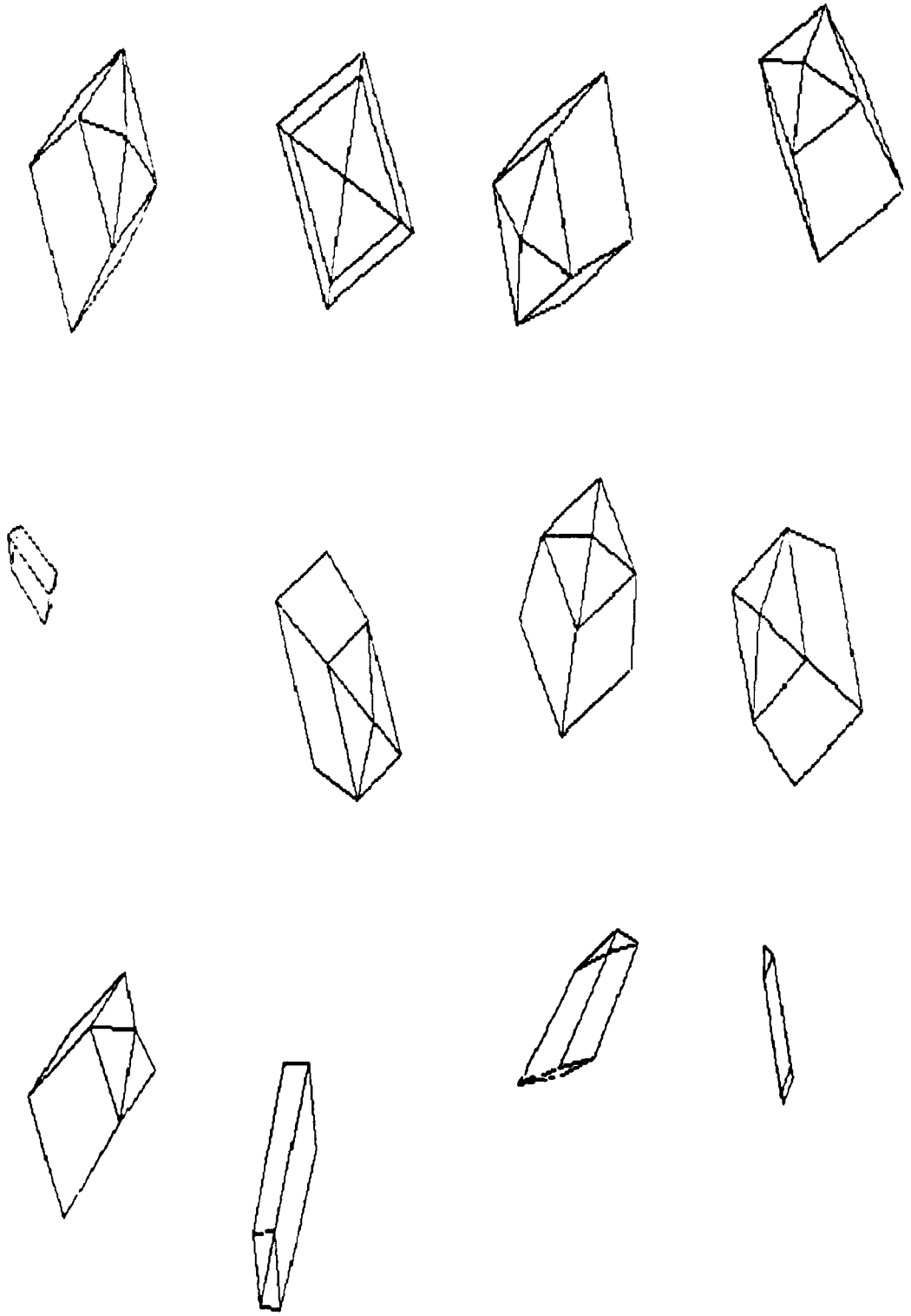


Figure 17

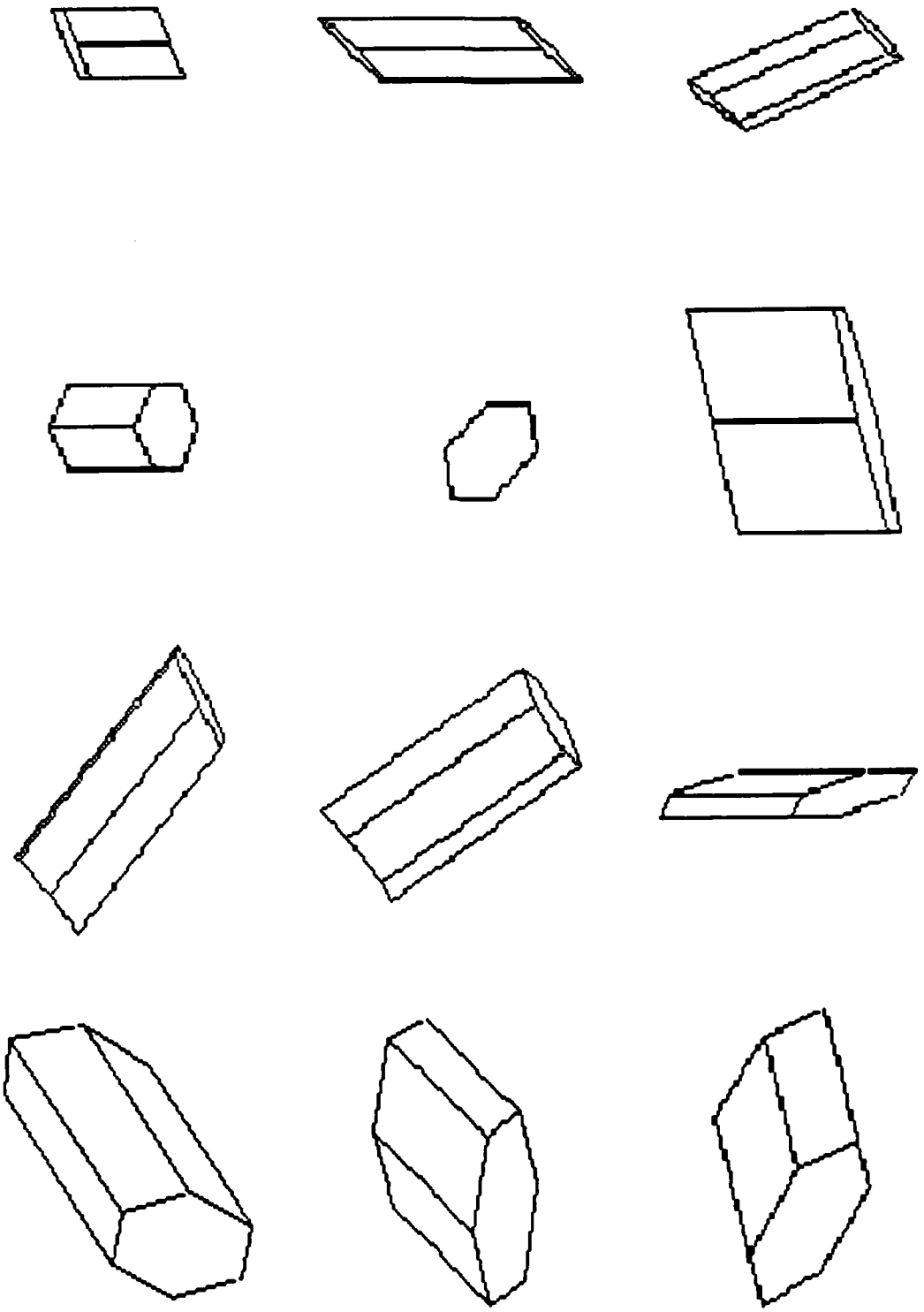


Figure 18

12. A Blocks World Recognition Algorithm: Using Principal Views

Suppose that there is a large set S of objects, which are all (convex) polyhedra. Suppose that this set of objects can initially be carefully studied. Many queries are anticipated of the form: Here is a line drawing. For what members of S is this a possible image? In the "offline" study and pre-processing phase, time is not of overwhelming importance. However, in the query phase, it is of extreme importance. (This is of course meant to be a simplified model of the real world recognition problem.) Suppose further (if your imagination has not been stretched too far) that for a particular polyhedron, each face has some kind of label (say color or letter) unique to that polyhedron but not to other polyhedra in S. An algorithm for processing the queries should quickly eliminate impossible candidates without undue effort. An outline of such an algorithm follows.

Let Q be a line drawing query.

Set C = S.

Set Pairs to be empty.

For each p in C

  If the set of faces visible in Q is not a subset of the faces of p

    remove p from C.

For each p in C

  If the set of faces visible in Q does not correspond to a set of faces

    in some principal view region of p, remove p from C

  else let r be the viewing region of p which corresponds to Q and

    add the pair (p,r) to Pairs.

For each (p,r) in Pairs

  If the Q is not topologically equivalent to a view of p from r,

    remove (p,r) from Pairs.

For each (p,r) in Pairs

  If the Q does not match (p,r) metrically, i.e., angles and distances

  are not possible for p, remove (p,r) from Pairs.

Set C = {p|(p,r) is in Pairs}.

After execution of this algorithm, C will contain all members of S for which Q is a possible image. It seems possible that in some applications the last computationally difficult step might even be omitted.

13. Orthographic vs. Projective Projection, or, Is it Really Worth Worrying about Projective Projection?

Typical discussions of principal views (under whatever name) only deal with orthographic projection. An important argument for this restriction is that there are many more views under perspective projection; perhaps too many to handle. There are two arguments in favor of considering perspective projection. One is that all equations become linear, thus making calculations much easier. The other is that, under some circumstances, views visible under perspective but not orthographic projection may be expected. For example, when a large object is encountered by a viewing system, the view obtained may well be of this kind.

A rough but useful approximation of the state of affairs in the real world with a given object and viewing system is this. The viewing system can't identify the object when the object is too close (it won't be in focus); neither can it detect the object when it is too far away. (We can't properly see an object when it is too close to our eyes; we also can't watch the butterflies ten miles away, even if they are in our line of sight.) Thus there is a roughly defined range that allows proper viewing. There are two fuzzily outlined "limit" spheres surrounding the object; the viewing system must be inside the outer sphere but outside the inner sphere in order to "see" the object properly.

This approximation helps to clarify an answer to the question: Can perspective *projection safely be ignored*? The view regions possible under perspective projection but not orthographic projection are precisely those of finite extent. All regions of infinite extent correspond to images possible under orthographic projection as well as perspective. Now if the inner limit sphere associated with an object and a viewing system properly contains all of the finite regions, they may be totally ignored; only orthographic projection is needed. If, at the opposite extreme, the outer limit sphere contains only regions of finite extent, then perspective projection absolutely must be considered. Of course, in many applications the situation is likely to be between these extremes. Call the imprecisely defined region between the limit spheres the *practical viewing space*; call the intersections of principal view regions and the practical viewing space the *practical viewing regions*. In an actual application, perspective projection ought to be ignored only if the practical viewing regions for non-orthographic views are empty or "small."

Under orthographic projection, it may be the case that some view regions on the viewing sphere have measure 0. Take the case of a cube. The viewing regions where only two faces are visible are great circles and the regions where just one face is visible are points. Such views are variously called degenerate or unstable in the literature. They seem to be highly unlikely. Can they therefore be ignored in the interest of efficiency? The limit spheres intuition is again helpful. There are no practical view regions of zero measure, but they might be quite small. Measuring the size of such regions compared to the total size of the practical viewing space should give an insight into the question of whether such views can be ignored.

One appropriate way to make determinations about ignoring some of the possible viewing regions is through probablistic calculations; exploring such considerations is beyond the scope of this paper. (See Swain (1988).)

14. Miscellaneous Thoughts on Principal Views

In the common definitions of equivalence relations, it is assumed that there is a label (such as a color) on each distinct face of the polyhedron and that the image will provide that label. This is not necessarily an unreasonable assumption. It is possible that texture, gray level, color, number of edges, or some other feature or combination of features of a face could be extracted from images. In that case, however, should we assume that every face has a distinct label? Or would it perhaps be more reasonable to assume that labels could be repeated on an object? Perhaps a cube is painted with two colors--would we want in practice to distinguish views that look exactly the same even though different faces are visible? This suggests yet another definition of the view equivalence relation.

Another question is this. Suppose it is anticipated that an image could easily contain inaccuracies. A very simple example is the case of a cube with one corner slightly nicked. A careful examination of such a cube and the vision system might indicate that in an image where this nicked corner is visible, the extra tiny face might or might not be detected. If the vision system somehow manages to produce a line drawing that either contains all three edges of the tiny face or none at all, the problem could be handled by storing views of the cube with the nick and without it. But would a real system be able to do this? Even if it could, suppose all corners of the cube are nicked, a couple of them doubly. Clearly there is a rapid explosion in the amount of storage and time required to handle such complications.

When considering the efficiency of an algorithm for calculating principal views, the following is worth noting. If they were to be used in the recognition of objects, the calculation of principal views would be part of a pre-processing phase. As in the block world example given above, actual queries--is this a such and so?--would presumably be made many times. Thus, when considering efficiency, the goal should not be to calculate the distinct views efficiently, but rather to store them in a data structure that does not take too much space, and against which efficient queries can be made. A subject for further research is the nature of such a data structure.

15. Conclusions

Clearly, most polyhedra in the real world are not convex. Any practical use of principal views of polyhedra requires consideration of the non-convex case. Discovery of an algorithm to deal with non-convex polyhedra (or at least a reasonably large subset of the non-convex polyhedra) under perspective projection is an objective for further research.

But, in the end, is this only a fascinating puzzle with no practical use? Do polyhedra just have too many principal views to admit useful computation? Consider now some simplifications that might be made.

1. Calculate the volume of each practical view region. If this value is very small or zero, omit the region from consideration. (Thus orthographic projection alone would be used for many objects, especially small objects.) This does require a decent estimate for the boundaries of the practical viewing region.

2. For objects that must always be in a stable position, i.e., not balanced on a vertex, consider only views of stable positions. (A building can't be balanced on one roof edge ...) This sort of approach is used in Chakravarty and Freeman (1982).

3. When the number of faces becomes large, make simplifications, grouping faces together. For example, consider a polyhedral approximation of a station wagon. It could consist of eight faces: front, back, two sides, bottom, top, hood, and windshield. (If it were not for accidents, chase scenes, and auto mechanics, we might also insist on stable positions only.) Another example is the description of a building. A shingled roof actually consists of many faces--but these need not be used as faces in a polyhedral description of the building. However, such a simplification must be done very carefully since it must be possible to determine the groupings in both images and the polyhedron with certainty.

4. Finally, some advantage might be taken of symmetry in highly symmetric objects.

In conclusion, it seems possible that using principal views in object recognition may be quite useful. An algorithm for determing principal views of a non-convex polyhedron under perspective projection is needed. Experimentation with actual applications will most likely be required to determine what sorts of simplifications (such as those just listed) are useful. Moreover, it may be the application that will determine the appropriate underlying equivalence relation.

16. Related Papers in the Literature

By far the most closely related other work this author has been able to find in the literature is that of Chakravarty and Freeman (see (Chakravarty and Freeman 1982) and (Chakravarty 1980)). Here, the equivalence classes of space are called vantage point domains and of images are called characteristic view partitions. Perspective projection is used and objects need not be polyhedral. The work is motivated by the idea of robot recognition of industrial parts. Of course, there are simplifications. It is assumed that all objects lie on a plane, called the supporting

plane. The objects are assumed to be in a stable position on the plane. The geometry of the camera plane relationship is fixed and the distance between the camera and object is in a limited range. Finally, a limited number of possible objects is assumed. From the examples given, it is possible that they must be fairly simple, but the objects are not necessarily polyhedral. A program was written to deal with this kind of recognition problem.

Edelsbrunner, O'Rourke, and Seidel (1983) discuss the idea of an "arrangement" of hyperplanes in d-dimensional space. The set of edge lines of a convex polygon in 2-space form an arrangement in 2-space as does the set of face planes of a convex polyhedron in 3-space. This paper includes the formula used to find the number of viewing regions as mentioned above. It gives an $O(n^d)$ time complexity algorithm for constructing a data structure representing an arrangement. However, this data structure does not explicitly enumerate the faces visible from each region; to do so requires $O(n^d + 1)$ space, and hence time.

In Edelsbrunner and Guibas (1986), arrangements are again studied and another algorithm is given. Here, each region is "visited" once using $O(n^d)$ time complexity. Since neither of these algorithms is designed to output an explicit list of visible faces for each region, they do not seem to be entirely practical for use in object recognition. Moreover, the data structures that are constructed contain information that is not required in object recognition. Nevertheless, the two papers may well contain fruitful ideas.

The recent paper by Plantinga and Dyer (1986) gives an algorithm for calculating the aspect graph for any polyhedron under orthographic projection. The nodes in the graph store a list of visible edges for distinct views of the polyhedron. The algorithm is complex and not all details are given. It seems to be the only paper which seriously deals with the non-convex case.

Werner, Baugher, and Gualteri (1986) deal with the "visual potential" (i.e., aspect graph) in the convex polygon case in exhaustive detail. They also include a useful section on related work. They give an $O(n^2)$ algorithm for constructing the visual potential.

In Asano et al. (1985), an efficient algorithm is given for dealing with the visibility of a collection of line segments in the plane such that the two segments can at most intersect at end points of the lines. The algorithm could be used for constructing the rays required for the non-convex planar case. It would also be useful in dealing with more than one polygon in the plane.

The author found no other useful papers which dealt directly with the problem as it has been defined here. That is not to say that there are none; since everyone seems to use different terminology, it is hard to be sure at all. There are numerous papers that deal with related problems that might shed insight, particularly in the development of good data structures. Here are a few of them. The classic paper (Huffman 1971) deals with the question: Can a given line drawing represent the

image of a polygon? More recently, Sugihara deals with the same question (1984a; 1984b). Suri and O'Rourke (1986) consider views from inside of a polygon. In Thorpe and Shafer (1983), constraints on changes in the line drawing of an object as a camera moves are discussed.

Acknowledgements

Invaluable insight and advice was provided by Chris Brown during the writing of this report. Drew Asson patiently did the programming that turned the abstract algorithm implementation described in Section 11 into a program that actually prints pictures of polyhedra. Peg Meeker skillfully prepared the text and figures for printing.

References

Asano, T., T. Asano, L. Guibas, J. Hershberger and H. Imai, "Visibility-Polygon Search and Euclidean Shortest Paths," *Proceedings, 25th Symposium on Foundations of Computer Science*, 1985.

Chakravarty, I. and H. Freeman, "Characteristic Views as a Basis for Three-Dimensional Object Recognition," *Proceedings, International Society for Optical Engineers*, Vol. 336, Robot Vision, 1982.

Chakravarty, I., "The Use of Characteristic Views as a Basis for Recognition of Three-Dimensional Objects," Technical Report IPL-TR-034, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, NY, October 1980.

Edelsbrunner, H. and L.J. Guibas, "Topologically Sweeping an Arrangement," *Proceedings, 18th STOC*, 1986.

Edelsbrunner, H., J. O'Rourke, and R. Seidel, "Constructing Arrangements of Lines and Hyperplanes with Applications," *Proceedings, 23rd Symposium on Foundations of Computer Science*, 1983.

Huffman, D.A., "Impossible Objects as Nonsense Sentences," *Machine Intelligence*, Vol. 6, 1971.

Koenderink, J.J. and A.J. van Doorn, "The Internal Representation of Solid Shape with Respect to Vision," *Biological Cybernetics*, Vol. 32, 1979.

Plantinga, W.H. and C.R. Dyer, "An Algorithm for Constructing the Aspect Graph," *Proceedings, 27th Symposium on Foundations of Computer Science*, October 1986.

Preparata, F.P. and M.I. Shamos. *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.

Stewman, J.H. and K.W. Bower, "Implementing Viewing Spheres: Automatic Construction of Aspect Graphs for Planar-faced, Convex Objects," *Proceedings, International Society for Optical Engineering*, Vol. 786, May 1987.

Sugihara, K., "An Algebraic and Combinatorial Approach to the Analysis of Line Drawings of Polyhedra," *Discrete Applied Mathematics*, Vol. 9, 1984.

Sugihara, K., "A Necessary and Sufficient Condition for a Picture to Represent a Polyhedral Scene," *IEEE Transactions on Pattern Analysis and Machine Intelligience*, No. 5, September 1984.

Suri, S. and J. O'Rourke, "Worst-Case Optimal Algorithms for Constructing Visibility Polygons with Holes," *Proceedings, 2nd Annual Symposium on Computational Geometry*, ACM, 1986.

Swain, M., "Object Recognition from a Large Database Using a Decision Tree," Thesis Proposal, University of Rochester, October 1987.

Thorpe, C. and S. Shafer, "Correspondence in Line Drawings of Multiple Views of Objects," *Proceedings, 8th International Joint Conference on AI*, 1983.

Werner, M., S. Baugher, and J.A. Gualteri, "The Visual Potential: One Convex Polygon," University of Maryland Technical Report, August 1986.

END

DATE

FILMED

5-88

DTIC